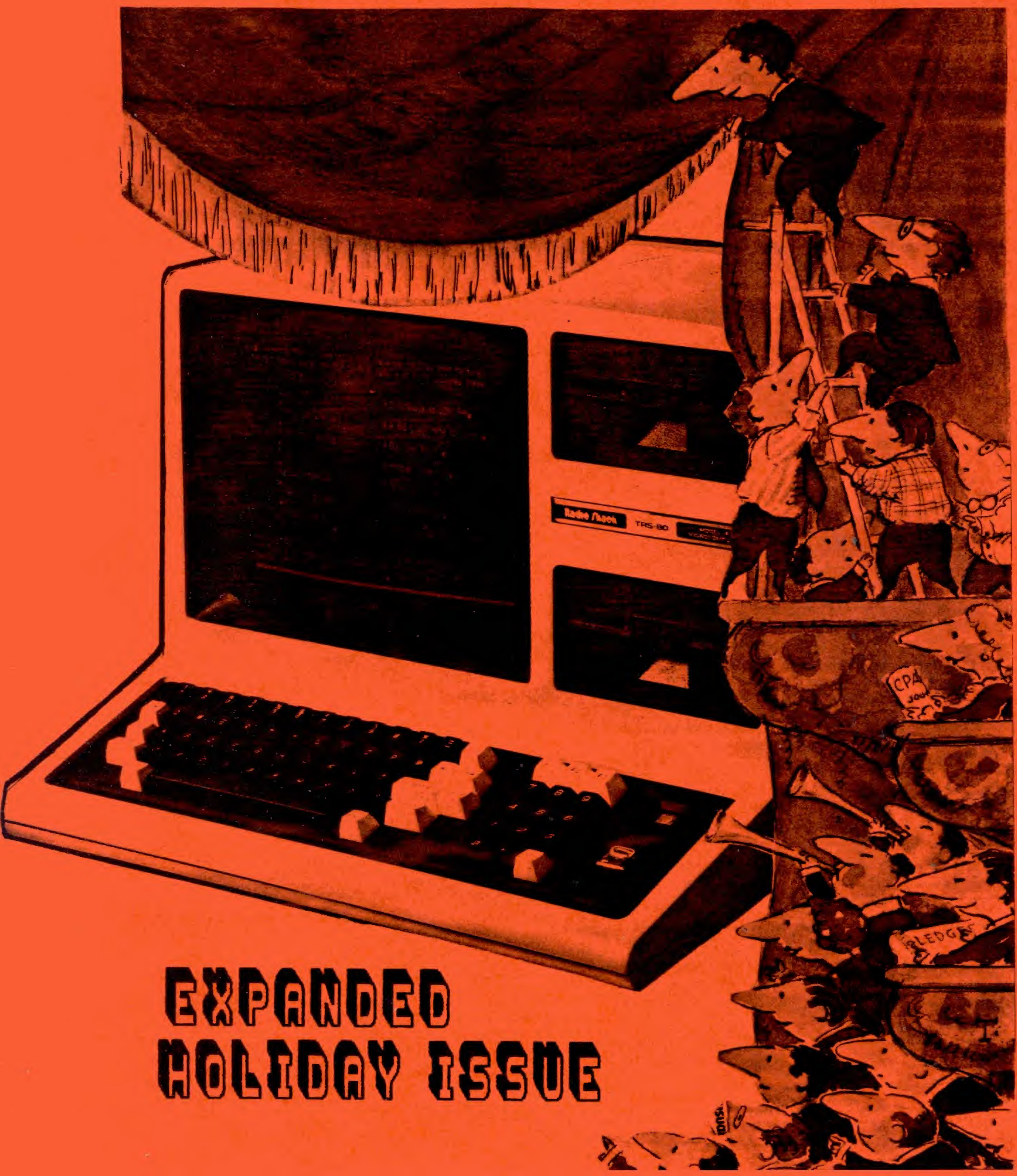
TRS TIMES

Keeping Models 3 & 4 Alive

Volume 1. No. 6. - November 1988 - $3.00



EXPANDED HOLIDAY ISSUE

LITTLE ORPHAN EIGHTY

It is absolutely amazing!

This column is written on October 16, 1988. This is exactly, to the date, one year after the people at 80 Micro made the fateful announcement that they would no longer support the TRS-80. The 'experts' didn't waste any time predicting that the machines would fade into oblivion to meet a quick and quiet death.

Boy, were they WRONG!

The TRS-80 has refused to roll over and die. It has proven to have a very dedicated following that is not about to junk a machine that does a commendable job day in and day out. We are as happy with it now as we were when we first brought it home from the store, and when the chips (pun intended) were down and the 'major-leaguers' threw in the towel for financial or other reasons, the 'little guys' took over.

The little guys? Yeah, I mean Stan Slater of Computer News 80, Luis Garcia-Barrio of TRSLINK, Tim Sewell of GEnie and the File Cabinet, and ourselves, TRSTimes. But we simply acted first. Had we not, I am sure that some other 'little guys' out there would have jumped in to fill the need.

What I am trying to say is that, with this kind of support, surely the TRS-80 has plenty of life left.

1988 saw more magazine coverage than anytime since March 1984. Computer News 80, which sported its first issue this past January, has grown into a respected and very fine source of timely information. Stan has done an incredible job. I urge anyone who is not at this time subscribing to do so. Let's keep this magazine around for a long time. Support the ones that support you.

TRSLINK, the monthly magazine on disk, put out their first issue in November 1987. Each new monthly issue has been better than the previous. It is obvious that Luis has put in many, many hours of his life to give the TRS-80 community support. Now, working long, hard hours is not particularly difficult if you get paid accordingly. However, TRSLINK is FREE. Nobody makes any money. His only motivation is LOVE of the TRS-80. So, to Luis and everyone else connected with TRSLINK, **THANK YOU**. May you long continue.

TRSTimes also started in January. At that time we had no idea just what we were getting into. I freely admit that some nights, usually around 2:30 am, it seemed like we wouldn't meet the deadlines. But, somehow, it always worked out. Such is the fun of publishing. We hope that we have brought you interesting articles and programs.

1988 also brought The FILE CABINET to the forefront of the TRS-80 scene. Tim has always been an avid collector of software, commercial and public domain. He has now what I believe to be the world's largest collection of TRS-80 software. Entering his computer room is an experience to treasure. I have never seen that many disks in my life. There are literally thousands, and Tim, with his infinite patience, is going through them, one at a time, to catalog the files into some reasonable order. Boy, oh boy!

Like they say: "It's a dirty job, but somebody's got to do it." and Tim is just the right man to undertake this giant and seemingly never-ending task. So give him your support. His work benefits us all .

Other fine contributions from the 'little guys' have surfaced. When our Model 4's would not accept dates after 1987, David Goben came to the rescue with his T62DOSXT package. Gary Campbell of DBSIDE enhanced Model III TRSDOS 1.3. so we can use our double-sided drives and, would you believe it, Ben Mesander has made available a C-compiler for the Model I.

A few of the 'big boys' are still around, creating magic for our machines. Most notably, of course, is Roy Soltoff. He is currently venturing to make IBM hard drives work on the TRS-80. I'll bet he will be successful and thus bring us reliable and inexpensive mass storage. I very much look forward to that.

I also hope that Roy will consider upgrading or rewriting (whatever) LS-DOS 6.3. to bring us the NEW STANDARD Model 4 DOS, **v6.4.**, in which the date function will accept dates for ALL years until the end of time, AND the alleged copy protection is removed, **whether it existed or not**.

Can we count you in on that one, Roy?

HYPERSOFT is still in the TRS-80 market. Their program, Hypercross, is the best file transfer utility available. Among the many other programs they offer is one for the IBM PC called PC-Four. It allows you to run Model 4 software on the IBM. Imagine running ALLWRITE on 'big blue'!

The latest rage for our machines is the HI-Rez board, which opens up a whole new world of computing. Micro-Labs are currently offering these at discount prices. Software is also readily available from them as well as from Microdex, who supports one of the best programs for HI-Rez, xT.CAD.

If you check the ads in CN-80, TRSLINK and TRSTimes, you will see that many other people are offering products of interest to the various TRS-80 machines. Check them out.

Looking back it is clear we are not quite as orphaned as the 'experts' would have liked us to believe a year ago. 1988 has been a good year.

Let's do it again in '89.

And now.....Welcome to TRSTimes #6.

TRSTimes - Volume 1. No. 6. - November 1988

CONTENTS:

TRSTimes is published bi-monthly by TRSTimes publications.
20311 Sherman Way #221. Canoga Park, CA. 91306. U.S.A.

1988 subscription rates (6 issues):
United States and Canada: $15.00 (U.S.)
All other countries: $20.00 (U.S.)

1989 subscription rates (6 issues):
United States and Canada: $18.00 (U.S.)
All other countries: $23.00 (U.S.)

THE MAIL ROOM

Comments on Issue #5

First off, I am very happy to hear that there will be another year of TRSTimes. I have been most impressed with the professionalism you brought to this venture, a level far in excess of that displayed by the late _80 Micro_. (As a contributor, I can attest that Lance goes to great lengths to avoid conflict of interest.)

However, there were a few little errors in TRSTimes #5 that I would like to respond to. Probably only someone with my nitpicking (some might say nitwit...) mind would notice them, but here goes anyway.

First off, in Roy Beck's CP/M column, he states that a TRSDOS 6.x JCL file cannot continue into an application; this is in error. I have used JCL's to MERGE BASIC programs, control the MEMDISK driver, and so on. One must be careful to plan out the prompts and necessary responses, but it can be done.

The next nitpick is in Gordon Collins' article on using SYS13/SYS. His first use of SYS13/SYS (copying VisiCalc to it so that it may be run simply by typing a "*" and hitting (ENTER)) is more accurately called an IEP: (I)nstant (E)xecution (P)rogram ; at least, that's what my documentation from Tandy calls that particular use. It's only when you alter EFLAG$ so that whatever is in SYS13 replaces the standard command interpreter (hence the designation ECI: (E)xtended (C)ommand (I)nterpreter) that the term ECI is appropriate.

Also, one needn't zap SYS0/SYS to make an ECI permanent; if you make the change in RAM with the MEMORY command, and then SYSGEN the system, the setting of the EFLAG$ will be preserved and restored on bootup.

Finally, a comment about Tim's PD Express. TrsDraw is written in Radio Shack's BasicG, and as such will not run under Micro-Labs' GBasic. HOWEVER, I am told by Micro-Labs owners that BasicG itself runs just fine on their boards; so, if you beg, borrow or (gasp!) steal a copy of BasicG, there is no problem with running TrsDraw on a Micro-Labs board.

From what I have found out, ALL the incompatibility between the two boards is in their widely divergent Graphics Basic; since one can run either Graphics Basic on either board, I'm now suggesting that people buy the Micro-Labs board (as it is much cheaper), and obtain a copy of BasicG so they can run programs written in either flavor of Graphics Basic.

So now that I've left the previous issue a pock-marked mass of ichor- dripping words and bleeding phrases (GRIN!), I'll raise my can of sodapop in toast to the coming year of TRSTimes!!

George Madison
Los Angeles, CA.

Issue #5 cover

Thank you for your support of the TRS-80. I'm glad to read that you will keep going for another year.

I wanted also to comment on the September issue. You are right in saying that it is your finest issue. The best part is the cover. TRSTimes will become famous for its loud covers, but this time there was a certain subtlety, a delicacy in the artwork. What I mean to say is that I noticed that two of the pictures on the front cover looked familiar. The picture on the center, bottom is a portion of a picture drawn by a good friend of mine, Ken Miller. He owns an Apple IIGS, so he used my machine to draw it. The picture of the river on the upper, right is by me. Ken is a sign painter and a commercial artist. I am not an artist, have never been an artist, might never be an artist. I feel flattered that you included my picture. I think it shows what computers can do for people. I write a lot more, now that I use a word processor. I draw, now that I have a Hi-Rez board. I'm not sure how, but the computer removes the intimidation of drawing. I like to sit and doodle now. Clearing the screen is even easier than tossing a sheet of paper. Erasing is another option that I like.

One other request. Could you give an explanation of how you put TRSTimes together? What hardware do you use; what software? I won't be too upset if you say that it was done on an IBM type machine. I hear that some people have been able to do productive work on theirs.

Peter Besenbruch
Ft. Leonard Wood, MO

Tim Sewell, very ably, did the cover for issue #5. He has thousands upon thousands of hi-rez pictures, so it was no easy task to select the right combination. I think his choices justly reflect the talent and ambition of the TRS-80 users. Our thanks go out to you and Ken for the use of your fine, well-drawn pictures.

TRSTimes is put together in a variety of ways. The front cover titles are done on ALLWRITE and Dot-Writer, run on Model III DosPlus. The rest of the cover, of course, is pasted up. Text submitted in Newdos/80 format is transferred to LDOS/TRSDOS6 format using COPYAID2. TRSDOS 1.3. text also is transferred to Model 4 format. It is then processed with LeScript.

Programs are tested on the DOS for which they are intended, and are then copied to Model 4 media in ASCII. A trial copy is then printed from LeScript.

Then comes the hard part: choosing the articles to get as much variety as possible, and the order in which they should appear.

When the material has finally been selected, it is then transferred to an IBM clone using either HYPER-CROSS or TRSCROSS, whichever is handier at the time. The material is then run into Ventura Publishing to shape the pages, taking advantage of the fonts and graphics capabilities for the final print-out; and then down the street to the printer for mass copies.

Accounting at TRSTimes is done on a VISICALC4 spreadsheet, backed up by using DAC EASY on the clone. Subscriber information is stored on Model 4 PFS:file. For the sake of safety, the same list is kept on the clone using Q&A.

Yes, the TRSDOS and MS-DOS machines can work together, and I feel that there is just something very appealing in the fact that the clone is actually promoting the word of TRSDOS.

-Ed.

Another junkie

I am glad you are going to publish TRSTimes through 1989. I enjoy your fine publication and manage to dig some goodies out of each issue.

I'm looking forward to you 'making me see the light to get me started' in assembly language. I don't have any burning desire to learn all the in's and out's, but sure would like to get to know it a little bit. Who knows, maybe it will become my new passion.. (The little lady says that is all I need, to have yet another passion with my computer.) "LIVING WITH A COMPUTER JUNKIE" is very good.

I have the Model 4, PRO-CREATE - Micro assembler, Editor Assembler Version 4, from MISOSYS. Maybe you could give us a clue in the November issue, just what software prerequisite we will need for your assembly language tutorials. We should get prepared you know.

Les Wolfe
Carson City, NV

A lot of our wives identified with Barbara Beck's article about Roy. When you're hooked, you're hooked.

See "CLOSE#6" for details on the AL tutorial.

-Ed.

MACROKEY

There is a potential bug in MACROKEY when used on a Model I. Programs that use high memory could lock up the keyboard. To fix this, insert line 365: **LD (4049),HL**

I really enjoyed this program. Keep up the great work. NEWDOS/80 rules!

Ben Mesander
Norman, OK.

CHANGO80

I found a bug in the listing to CHANGO80/BAS from issue #4. Line 340 is missing a 'greater than' and a 'less than' sign. Line 340 should read:

```
340 IF I<1 OR I>8 THEN 330 ELSE
LO=642+((I-1)*10):GOSUB 40:ON I
GOTO 480,470,460,450,440,430,420,410
```

I enjoy the type-in programs, especially if I find - and can correct mistakes. Makes me feel like a programmer. Keep the CP/M series coming. Also liked Fred Blechman's article on buying and selling used computers. Sign me up for '89.

Torben Birkholm
Aarhus, Denmark

You caught one Ventura Publisher's peculiarities. It will all too often drop the 'greater than' and 'less than' characters. I thought I got them all. Sorry.

Glad to have you with us in '89.... SKAAL!

-Ed.

SUPERSCRIPSIT

Here is a hint for users of Model 4 Superscripsit. I have detected that a lot of the troubles with that program arises due to the 'type-ahead' when editing. I turn off the 'type-ahead' and avoid all kinds of mishaps when working over a document.

SYSTEM (TYPE=NO) does it.

A second comment: I have outboard disk handlers where I sometime get error messages like 'Directory read error' when I **CAT :2** from DOS. Go to Basic and ask for CAT :2 through the system with LS-DOS 6.3, for example: **SYSTEM"CAT :2"**. This usually rolls the directory right out with no fuss. The file names aren't in alphabetical order, but it goes and allows me to find the proper file names to COPY

out to a new disk before the old one get so bad it is lost.

Now, for the hackers.... When looking through the 'SOURCE' I notice that the Z-80 instruction is always in the form of RST 28H for SVC call to the system. Are any other addresses used in LS-DOS? It might make a nice tutorial for TRSTimes some day if anyone knows the rationale for using any special value of address location in the system.

I have a bare, older Model 4 with 64K and two single sided drives. My software includes Profile, Multiplan, Superscripsit, ALDS, Newdos/80 and DOSPLUS4.

Edward F. De Mers
El Cajon, CA.

Thanks for the Superscripsit and CAT tips.

The whole purpose of SVC's (SuperVisor Calls) is to make life as easy as possible for the author of the DOS. Model I & III DOS calls were made to specific addresses, which created a lot of hate-mail when the DOS was updated and in the process, some addresses were changed. No longer are specific address calls necessary, instead the A register is loaded with the SVC number, followed by RST 28H to execute the call (depending on the routine, other registers may also need to be loaded with specific values). Thus, anyone using the SVC's can be assured that their program will run on TRSDOS 6.2., as well as LS-DOS 6.3., and for the most part, also on DOSPLUS 4.

However, being hackers, we will present a list of specific CALL addresses for TRSDOS 6.2. in a future issue.

-Ed.

Canadian comments

Thank you very much for TRSTimes! You are the first to publish the famous "ONE BYTE PATCH"; the un-lamented 80 Micro kept such useful information from us. I particularly liked your article about TRSDOS 1.3. I prefer to induce PASSWORD-amnesia in my operating systems, rather than using SU+ to strip passwords from my disks.

My computer is a 4 drive, 128K Model 4. I use mostly LDOS 5.3. and LS-DOS 6.3. Northern Bytes were great for surgery on 1.3., and David Goben published some patches for TRSDOS 6.2. The patch for the FLAG byte in LS-DOS 6.3. is the same, but **D02,50 = 18** is the one for SYS2.

So far, I have not been able to get into LDOS, and using SU+ messes up the date field, but I can live with that. I have never had the chance to use Newdos and don't intend to either. APPARAT deserted us long before Radio Shack did, and there is no MS-DOS computer in my future, particularly not of the Tandy variety. With the cost of RAM gone into the stratosphere, any plans for memory expansion of my old 4 are on hold. My computer was one of the last Model 4's produced, and it let's me choose between 7 and 8 memory refresh cycles. Maybe when the price of 256K chips returns to normal I might try Roger Alford's method for using 256K chips in the second bank.

As a hardware hacker of almost 50 years standing, I was having some trouble expanding to 128K. I had made it a point to use the Tandy chips specified in the tech manual, and I could format and list a Ramdisk. But when I wrote to it and attempted to use the program, all I got was garbage. With RAM being cheap at the time, I bought another set of chips with the same result. Upon closer inspection, my originals looked very much like 8 cycle refresh RAMs (4164). So now I had 128K 7 cycle RAM and 64K 8 cycle RAM. Never one to waste a penny, I set to work. There was this jumper (J1) right near the memory matrix, and out it came. All the memory was replaced with 7 cycle, and my Model 4 has performed like a dream ever since.

I am looking forward to your series about assembly language programming since the nitty-gritty of programming has so far eluded me.

Happy computing.

Willi Wald
Hamilton, Ontario
Canada

I don't think 80 Micro ever 'withheld' information from us. Wayne Green certainly published any and all information available. However, after his departure, the direction and the contents of the magazine changed drastically. It turned away from the hobbyist to cater to the general user. It is quite possible that certain items were not published, either because the editors considered it above the readers heads, or maybe they were just playing it safe for one reason or another. This was obviously not successful.

As stated in Little Orphan Eighty column from the January 1988 issue, TRSTimes will try to emulate the Wayne Green years as much as possible. In other words, if we can get a hold of useful information (and permission from the author), we'll publish it.

Many people have contributed patches to the various DOS's. You mention Northern Bytes and David Goben. Don't forget about Mr. Patch himself, Andy Levinson. Also, Henry Herrdegen, from Windsor, Ontario, just did a large series of TRSDOS 1.3. patches in the September issue of Computer News 80. Check out his fine article there. They also have a disk available with all the patches.

-Ed.

The TRSTimes thought for the day:

MS-DOS..... just say NO!

The Real Hackers

by Eric Bagai

We've read the stories about Woz, Shrayer, Gates, and the others. But they were not real Hackers any more than Marconi and Edison were real Hams. They were merely the first. The real Hackers came after them.

In the early eighties you could depend on a half-dozen articles a month titled "Why Johnny Can't Compute," or "Computer Literacy: Another Failure in the Schools." These were written not by computer experts, but by the people who wrote yesterday's news and entertainment, and who will write tomorrow's. Their primary effect was to sell copy and terrify parents. The real influence of the microcomputer on the young was more varied, and much more interesting.

From the very first, some of the kids found that they could outclass the teacher and gain recognition if they could hack a decent line of code. While their teachers were taking night classes in PILOT and LOGO (two credits each toward the next salary step) or listening to a Honeywell-trained MIS director talk about data integrity, the kids were learninghow to screw up the next kid to sit down at that keyboard. They learned random seeding and graphic string packing to make a faster version of Astro. They learned how to use symbolic logic, and multidimensional arrays and sorts to hide the clues in their adventure games. They traded peek and poke points like baseball cards. They became competent. They became Hackers.

Most of their more normal age-peers had no incentive to learn something that adults couldn't do, that wasn't in the textbook, and whose rewards were uncertain. Most kids were lucky enough to have their competence in skills that the system recognized, and were satisfied with that. I certainly don't contend that the average child never hacked, or that only the strange ones did. But more often than not the local Hackers were not the more socially adept, academically successful, or junior achiever-type kids. This mild truism was exaggerated to produce the stereotype of the Computer Nerd.

Stereotyping an exception to the norm is not a new phenomenon. Much the same thing happened a few generations ago, when the automobile was new. Maybe Ford made it, but only the weird kid down the block could fix it so it stayed fixed. And more likely than not he wasn't doing all that well in school. The handy (and just as untrue) stereotype for these kids was Grease Monkey. The schools quickly adopted this attitude as gospel, and taught future auto mechanics in the shop classes, where you got dirty and weren't on the academic track.

For the enthusiast, microcomputers and automobiles are much alike. They both are immediately reinforcing and, when the work on them is done, provide entertainment and garner praise. They both hold out the mythic goal of creating the next "PacMan" or "Visicalc", or of constructing the car that blows away the "Baja" or attracts the attention of GM. They are also both non-literate hobbies, at least at the early stages.

During the early days of microcomputers, if you read the README1.ST or /DOC files, or the REMark statements in the programs of the day, you discovered that the programmer who was so fluent in BASIC or assembler often had no sense of English grammar, syntax, or spelling. Style was one's use of exclamation points, or the catchy phrase you signed off with. The New Literacy was also the Different Literacy; and for some, obviously, it was the Only Literacy.

This brings up an interesting side-effect. Some kids learned to read with the Thompson's auto repair manuals. Others learned to read with BASIC Better and Faster, or the documentation to Turbo-Pascal. For many kids, learning to read is the result of a need that has nothing to do with school. And, then as now, some kids just need to gain the confidence that comes with true competence before they can accept or submit to the discipline of school.

At the beginning of this century kids were moving up from the crystal sets their parents unwittingly gave them to the amateur radio bands. They learned Morse code, listened to the beeps and clicks between the few commercial radio stations, and began to recognize the "fists" that came on regularly. Their first CW key was a drawer knob and two strips of copper, mounted on the back of a rat trap. Their first broadcast CQ was probably illiterate and certainly illegal, but it was heaven. Some of them became competent enough at building radios out of scrap (and repairing them with more scrap) to get jobs as radio operators on the great ocean-going barges during the first third of the twentieth century. They were the proud masters of the only weatherproof structures

built atop the huge flat decks: the original Radio Shacks.

Because their Shack was the highest point on the ocean, and their whip and dipole antenae even higher, they were often the literal center of every passing lightning storm -- which is why their generic name was Sparks. The stereotype with which they were inflicted was of the excitable red-headed kid who could understand static and warn of pending disasters. His most recent incarnation was as Corporal Radar O'Reilly, in M.A.S.H.

Every generation can be characterized, not only by its most typical aspects, but by the alternatives to the mainstream that it produces. For a few members of this generation that alternative was microcomputer Hacking. These are not the software pirates, the system crackers, or the phone phreakers -- criminals can be found in any group. They are the Hacker enthusiasts who came to love microcomputers not just for what they could do, but for their absolute honesty and infinite patience.

Eventually society incorporates its alternative styles into the larger images of an age, sometimes exaggerating their size (e.g., the Hackers) or deprecating their influence (e.g., the Hippies.) Very soon computers will be enough a part of the school curriculum so that good programming will merely elicit good grades. This does not mean that there will be no more Hackers. Just as, every day, some kid discovers the secrets of Morse code or the wonders of the internal combustion engine, another kid discovers that microcomputers, unlike their teachers or parents, do not judge them or make false promises. There will always be Grease Monkeys and Hams and, even though their day is almost gone, there will always be Hackers.

What matters now is finding the next wave. I'm sure it will be terrifying and fun and misrepresented by the press. It will probably be something that requires a young, flexible mind.

Something like teleportation!!

• Model 4 · Hi-Rez

TRSTEXT2

(The Next Generation)

by George Madison

According to our Gentle Editor, my program TrsText (published in the May issue of TRSTimes) has generated positive feedback, a bit of news that warms my Hacker's heart. It's a great feeling to be told you've created something that people find useful.

This time, the announcement isn't quite so radical. You're used to the idea that by adding TrsText to TrsDraw, you can now use your DotWriter fonts on your HiRes screen. All this article does is describe a minor, little, tiny tweek to the program.....

which speeds it up about 650%!

Actually, there's more to it than that. I've added two "user friendliness" features as well as the speedup. And so that those who haven't already installed TrsText don't have to go digging for the May issue, the entire listing for TrsText2 is printed below, as well as the Assembly listing for the machine language subroutine, for those of you who are interested in how it works.

WHAT'S THE DIFFERENCE?

You may be wondering how I managed to speed the program up by more than six times. The answer is quite easy: more of it is written in Assembly Language now.

The very first version of TrsText was ALL BASIC... and it was VERY slow. It did work, however, and that was quite encouraging. Being in BASIC, it had to use exponentiation (a very slow function in BASIC) to access the individual pixels in each byte of DotWriter data for transfer to the screen. The version that was printed in TRSTimes used a short Assembly subroutine to handle this. A byte of data and the starting coordinates were passed to the routine, and it did the rest. Since bit-fiddling is much easier and much FASTER in Assembly than in BASIC, that one little routine made TrsText run about 4x faster than the original, all-BASIC version, lifting it from being a curiousity into actual usability.

Time passed, and I got to thinking about the TrsText routines again. It seemed inefficient to me to have to fetch and shuttle the data a mere one byte at a time. On the other hand, there was certain information (character width, number of print lines, etc.) that had to be gotten from the DotWriter file, and the easiest way to get it was as a random access file with a one-byte Logical Record Length. There was also the proportional width table at the end of the font file to think about. So, I thought.

What the new TrsText2 does is to open the font file with a 1-byte LRL, extract the character size data, and preserve in an array the entire proportional width table. It then closes the file, and reopens it with an LRL that allows TrsText2 to grab data in blocks the maximum width of the character. The rewritten Assembly routine can accept this block of data "whole", speeding up the throughput. I would imagine also that not having to go to the end of the font file to retrieve the width data between every character also adds its bit of speed increase.

The only problem now is that it does take somewhat longer to get to the point where you can tell TrsText2 what you want it to print on your screen. This wouldn't be so bad, except that if you type in more than will fit, you were kicked back to the main TrsDraw screen, and had to start all over again, from scratch. FRUSTRATING! So, in the new TrsText2, you will simply receive a message that what you have typed will not fit on the screen, and you are given a chance to try again. Much nicer, and saves you time.

The other problem was the fact that although TrsDraw was written to be able to show you directories of your /HR picture files, it was never intended to do the same for your /PR font files. Given the sometimes cryptic names of DotWriter fonts, it's easy to forget the exact name of the font you want to use. So now, if you type ? <ENTER> at the the font name prompt, you will be presented with a directory of all the fonts online at that time. You are then asked again for the name of the font you want to use -- with the directory still on the screen, to make your selection easy!

INSTALLATION

Just the same as for the previous version. Type in TRSTEXT2/BAS as you see it below. Save it in ASCII format with the command:

SAVE"TRSTEXT2/BAS",A

Load your WORKING copy of TrsDraw (never, EVER modify your master copy!) and give the command: **MERGE"TRSTEXT2/BAS"**

You can now save the modified program back to disk, and enjoy the additional convenience and improved speed!

ACKNOWLEGEMENTS:

Like the previous version, TrsText2 is based on information about the structure of DotWriter files provided by Scott McBurney (S.MCBURNEY on GEnie).

```
6 CLEAR:DEFINT A-Z:CLS:SCREEN 1:
OUT 142,1:SYSTEM "system (break = n)":
RESTORE 151:FOR N = 1 TO 116:READ A$:
FO$ = FO$ + CHR$(VAL("&H0" + A$)):NEXT

8 DIM X,Y,U,V,XX,YY,FL,FG,C,N,M,PN,SX,PT,SF,
B1,B2,SZ,UU,VV,AD!,AR!,A$,B$,AR$,SS$,CM$,
C2$,FS$,DD$,DC$, RV$,BK$,CU(5),MK(5),ER(3),
ES(3),CT(2),CS(4),TH(5),TV(10),TT(10),T$(23),
TX(23),TY(23),X.(25),Y.(25),FB$,FH,FF,PR(96)

12 DIM SN(825):IF MEM < 512 THEN :PRINT:
PRINT"Insufficient memory";CHR$(14):
SYSTEM"system (break = y)":OUT 140,0:
OUT 141,0:OUT 142,0:VIEW(0,0)-(639,239):
CLEAR:SYSTEM ELSE ERASE SN

43 IF CU = 1 THEN IF FL = 2 OR FL = 6 OR FL = 13
OR FL = 20 OR FL = 24 THEN
PUT(X-3,(Y*ABS(FG0)) + V-3),CU:
PUT ((X*ABS(FG0)) + U-3,Y-3),CU:
PUT((X*ABS(FG0)) + U-3,(Y*ABS(FG0)) + V-3),CU

44 WEND:IF CC = -1 AND FL < > 5 THEN
PUT(X-3,Y-3),CU:IF FL = 2 OR FL = 6 OR FL = 13
OR FL = 20 OR FL = 24 THEN
PUT(X-3,(Y*ABS(FG0)) + V-3),CU:
PUT((X*ABS(FG0)) + U-3,Y-3),CU:
PUT((X*ABS(FG0)) + U-3,(Y*ABS(FG0)) + V-3),CU

61 M = 0:IF FL < > 0 THEN ON FL GOTO
36,95,100,161,126,85,200,66,68,36,36,90,115,64,
36,74,36,36, 69,139,36,36,80,36,36,36

119 GOSUB 35:IF SZ > 825 THEN
SOUND 0,0: RETURN ELSE DIM SN(SZ):
GET(X,Y)-(U,V),SN:U = U-X:V = V-Y

129 IF B = 20 THEN SCREEN 1:
PRINT "Enter the DotWriter Font filename you wish
to use:";: LINE INPUT FONT$:IF LEN(FONT$) = 0
THEN CLS: SCREEN:GOTO 42:ELSE IF FONT$
= "?" THEN SYSTEM"CAT /PR":GOTO 129

130 IF INSTR(FONT$,"/") = 0 AND INSTR
(FONT$,":") < > 0 THEN FONT$ = LEFT$
(FONT$,INSTR(FONT$,":")-1) +
"/PR" + RIGHT$(FONT$,LEN(FONT$)-INSTR
(FONT$ ,":") + 1)

131 IF INSTR(FONT$,"/") = 0 THEN
FONT$ = FONT$ + "/PR"

132 INPUT"Input # of dots between characters
(0-9)";SD: ON ERROR GOTO 150

133 OPEN "I",1,FONT$:CLOSE:
OPEN"R",1,FONT$,1:FIELD 1,1 AS C$:
ON ERROR GOTO 0

134 GET 1,1:L = ASC(C$):GET 1,2:
L = L + 256*ASC(C$):GET 1,3:W = ASC(C$):
GET 1,9:NC = ASC(C$):IF NC > 255
THEN NC = 255

135 GET 1,7:PL = ASC(C$):IF PL > 6 THEN PL = 6

136 FOR I = 1 TO 96:PR = L*NC + I + 1:GET 1,PR:
PR(I) = ASC(C$):NEXT I:PR(0) = INT(W*.6):
CLOSE:OPEN"R",1,FONT$,W:FIELD 1,W AS C$

137 LINE INPUT "Text:";CHAR$:
IF LEN(CHAR$) = 0 THEN CLOSE:
CLS:SCREEN:GOTO 42

138 U = 0:FOR I = 1 TO LEN(CHAR$):
A$ = MID$(CHAR$,I,1):GOSUB 149:
U = U + W1 + SD: NEXT I:U = U-SD:IF U > 640
THEN PRINT"Line too long to fit!":GOTO 137:
ELSE CLS:SCREEN:V = PL*8:FG = 20:FL = 20:
GOSUB 32:GOTO 42

139 IF B = 28 THEN U = 0:V = 0:FG = 0:FL = 0:
CLOSE:GOTO 42:ELSE IF B < > 27
THEN GOTO 42

140 FO! = VARPTR(FO$):
FO! = PEEK(FO! + 1) + PEEK(FO! + 2)*256:
FOR I = 1 TO LEN(CHAR$):
A$ = MID$(CHAR$,I,1):GOSUB 149

141 Z = 1:IF A$ = " " THEN X = X + W1: GOTO 148

142 A = PL*(ASC(A$)-32)

143 FOR J = 0 TO PL-1

144 GET 1,A + J + 1

145 DY = (Y + J*8):D$ = LEFT$(C$,W1): CALL FO!
(X,DY,D$)

146 NEXT J

147 X = X + W1 + SD

148 NEXT I:FG = 0:FL = 0:CLOSE:GOTO 42

149 W1 = PR(ASC(A$)-32):RETURN

150 IF ERR < > 0 THEN CLS:
PRINT"Font ";FONT$;" is not on-line; please try
again!":PRINT:RESUME 129

151 DATA DB,83,F5,3E,73,D3,83,C5,D5,
E5,DD,E1,DD,6E,00,DD,66,01,DD,E1

152 DATA DD,4E,00,DD,E1,DD,46,00,DD,
5E,01,DD,56,02,D5,DD,E1,E5,FD,E1
```

```
153 DATA C5,7D,E6,07,CB,1C,CB,1D,CB,
1C,CB,1D,CB,1C,CB,1D,67,3E,07,94

154 DATA 47,3E,01,28,04,CB,27,10,FC,
5F,55,C1,79,D3,81,7A,D3,80,C5,06

155 DATA 08,DD,4E,00,CB,01,DB,82,38,
07,67,7B,EE,FF,A4,18,01,B3,D3,82

156 DATA 10,EE,DD,23,FD,23,FD,E5,E1,
C1,10,B8,F1,D3,83,C9

157 '

158 ' TRSTEXT/BAS was written by George
Madison, based on concepts by

159 ' Scott McBurney (S.MCBURNEY on GEnie).
TRSTEXT is (c) 1988 by

160 ' George Madison.
```

TRSTEXT2/ASM

```
; TRSTEXT2/ASM -- A routine to assist in FAST
; drawing of DotWriter fonts on the HIRes screen.
;
SETSCR  IN    A,(131) ;Grab current HIRes
                      ;screen status
        PUSH  AF      ;...and preserve it
        LD    A,73H
        OUT   (131),A;Set HIRes screen
                      ;status to our liking.
PROCARG PUSH  BC      ;Save passed parameters
                      ;on the stack so we can
        PUSH  DE      ;process them one at a time.
        PUSH  HL
        POP   IX
        LD    L,(IX)
        LD    H,(IX+1) ;HL = X Coordinate
        POP   IX
        LD    C,(IX)  ;C = Y Coordinate
        POP   IX
        LD    B,(IX)  ;B = Length of DotWriter data
        LD    E,(IX+1)
        LD    D,(IX+2) ;DE = Pointer to
                      ;DotWriter data
        PUSH  DE
        POP   IX      ;Move data pointer to IX
        PUSH  HL
        POP   IY      ;Copy X coordinate to IY
CNVCORD PUSH  BC      ;I got this routine to
                      ;convert the X coordinate
                      ;to a number and bitmask
        LD    A,L     ;from the documentation to
        AND   7       ;the HIRes utilities XINIT,
                      ;XCUT and XPASTE, which
        RR    H       ;were written by
                      ;Paul Bradshaw.
        RR    L
        RR    H
        RR    L
        RR    H
        RR    L
        LD    H,A
        LD    A,7
        SUB   H
        LD    B,A
        LD    A,1
        JR    Z,SKIP
LOOP    SLA   A
        DJNZ  LOOP
SKIP    LD    E,A     ;E = Bitmask for the proper
                      ;pixel within byte
        LD    D,L     ;D = Byte the pixel is in
        POP   BC
SETIT   LD    A,C
        OUT   (129),A ;Set Y coordinate
        LD    A,D
        OUT   (128),A ;Set X coordinate
        PUSH  BC      ;Preserve Y coordinate
                      ;& data length
        LD    B,8     ;8 bits of data per byte
        LD    C,(IX)  ;Get the DotWriter data
                      ;from the buffer
START   RLC   C       ;Rotate leftmost bit
                      ;into carry flag
        IN    A,(130);Get existing graphics
                      ;screen data
        JR    C,SET   ;Carry flag set?
                      ;Turn the pixel on!
        LD    H,A     ;If not...
        LD    A,E
        XOR   0FFH    ;We invert the bitmask....
        AND   H       ;And turn the pixel OFF!
        JR    NEXT
SET     OR    E       ;Turn the pixel on
                      ;using the bitmask
NEXT    OUT   (130),A ;Return the modified byte
                      ;to the HIRes screen
        DJNZ  START   ;...and keep going until we
                      ;finish this byte.
        INC   IX      ;Point at the next byte
                      ;of DotWriter data
        INC   IY      ;Increment the X coordinate
        PUSH  IY
        POP   HL      ;and copy it to HL for
                      ;CNVCORD to work on.
        POP   BC      ;Restore the data length counte
        DJNZ  CNVCORD ;...and keep going until
                      ;we're out of data.
        POP   AF      ;Get the previous screen status,
        OUT   (131),A ;Reset the status,
        RET           ;And return to BASIC.
        END   SETSCR
```

Write to George Madison with your questions or comments - either c/o TRSTimes - or directly to:
2929 Waverly Drive #212
Los Angeles, CA 90039-2033

• MODEL III

TRSDOS 1.3. CORNER

by Lance Wolstrup

TRSDOS 1.3., the granddaddy of Model III Disk Operating Systems, is still being used by a large number of TRS-80 users. It is a good DOS, though it lacks some of the fancier features of its competitors.

Over the years some of its shortcomings were fixed by patches written by various experts, most notably Andy Levinson in his series of articles in 80 Micro. Computer News 80 published patches by David Goben and in their September 1988 issue, Henry Herrdegen presented a fine collection of most of the known patches. I contributed a few patches myself in our last issue.

I assume that both David's and Henry's articles generated a lot of mail for CN80. Certainly, TRSDOS 1.3. CORNER of last issue produced our largest mail response to date. What I am trying to say is that the interest in 'fixing-up' TRSDOS 1.3. is there. It seems as though this DOS is enjoying a revival.

Both my Model III and 4 originally came with two single-sided drives. I soon found that this was not enough, so I bought two more drives. Oh, it was heaven - at least until double-sided drives became available and relatively cheap. You guessed it: I upgraded, of course.

Some of you got double-sided drives as original equipment when you bought your Model 4. Some jumped on the bandwagon, as I did, and sprung for the extra disk space. This works wonderfully when using either a Model 4 DOS or one of the fancy Model III DOSes. However, if you are one of the many who uses TRSDOS 1.3., your double-sided drives are being thoroughly wasted.

The problem:

TRSDOS 1.3. is only capable of reading one side.

PATCHER

This problem has been solved once and for all. Gary Campbell, a very talented programmer from British Columbia in Canada, sat down and wrote an extensive machine language program that modifies TRSDOS 1.3. so it will accept and use double-sided drives. This is not just a series of patches, it is an extensive rewrite of DOS itself.

The program is called 'PATCHER' and it does remarkable things. Not only does it allow access to double-sided drives, it also gives you four new LIBrary commands, as well as fixing all known bugs in DOS. Most of the patches mentioned above have been incorporated and, best of all, YOU DON'T HAVE TO DO IT. The program does it for you - it's easy and very fast.

MAKING TRSDOS 1.4.

First, make a copy of your original, unmodified TRSDOS 1.3. master disk.

When done, reboot with your new copy in drive :0. Insert the PATCHER disk in drive :1 and type **PATCHER** <ENTER>.

You will be presented with a front screen telling you to use only a copy of an unmodified TRSDOS 1.3. disk. You can press any key to go on.

The next screen tells you basically the same thing. Pressing <ENTER> makes PATCHER go to work. You will see various tracks and sectors being rewritten. In just a few seconds the whole thing is done, and the screen tells you to be sure and read the documentation /TXT files included on the disk.

Pressing <ENTER> reboots the system and, WOW, you now have TRSDOS 1.4.

At this point use the first of the four new LIBrary commands, type **DBSIDE** <ENTER> and you are in business.

Yes, this is a true upgrade. You now have a better, more flexible DOS to work with, something that Radio Shack would probably have done themselves had they continued support for Model III.

WHAT'S NEW?

"Just what is better about it?". Well, I'm glad you asked. For starters, of course, the double-sided drives will now read both sides. Here I must mention that the two sides are NOT treated as one large one, as is standard on the other DOSes; instead they are treated as two individual single-sided disks. In essence, you have just doubled the number of disk drives in your system.

If you have 2 double-sided drives, TRSDOS 1.4. will access them as 4 single-sided drives, or if you have 4 DS drives, you now have 8 SS drives.

Now, wait a minute. We all know that TRSDOS 1.3. cannot use more that 4 drives! True, but remember, this is no longer TRSDOS 1.3., it is now TRSDOS 1.4., and it can read up to 8 drives.

The drives are numbered as follows:

Drive 0 - top side = 0 bottom side = 4
Drive 1 - top side = 1 bottom side = 5
Drive 2 - top side = 2 bottom side = 6
Drive 3 - top side = 3 bottom side = 7

In other words, if you have 2 double sided drives, you will now be able to access drive 0,1,4, and 5. While this does take a little getting used to, it becomes second nature in no time at all.

As mentioned above, four new LIBrary commands are now available for your use. The first, **DBSIDE**, should be used immediately upon boot-up to install the system. The others are:

BOOT - this command resets the system to original default conditions.
CAT - displays a short directory of visible files on specified drive. Do NOT use a colon with this command.
SWAP - swaps one drive for another.

While BOOT and CAT are both self-explanetory, SWAP needs some clarification:

The correct syntax for SWAP is: **SWAP (dn,dn)**

dn, of course, means drive number. For example, if you have two drives, you may not like the fact that DOS thinks of the bottom sides as drive 4 and 5. Use the SWAP command to force DOS to recognize the bottom sides as 2 and 3 by:
SWAP (4,2) <ENTER>
SWAP (5,3) <ENTER>

Some commercial software will balk at drive numbers larger than :3, so this command becomes very useful. You need not attempt to PATCH the software, simply SWAP in your drive with the data disk. Imagine the possibilities!

FORMAT and BACKUP have been changed extensively to access up to 8 drives correctly. The master password is no longer used. (bye-bye 'backup limited diskettes'). FORMAT allows you to verify as usual, or skip verification, which speeds up the process immensely.

The directory display has been improved to show the full date on all new files. Old file dates will, unfortunately, look somewhat strange. However, that is a minor trade-off.

The DIR command no longer scrolls the filenames off the screen; instead, if there are more files than the screen can display, it shows them one screenful at a time.

The LIST command has been changed to display in ASCII as the default.

The annoying 'master password' prompt has been eliminated from the PURGE command.

THE DOWNSIDE

I would be remiss in my duty as a reviewer if I did not report the one thing that I did not particularly like about TRSDOS 1.4. The documentation plainly states that only the SYStem files are copied over to your new master disk. This is all too true.

Your new master disk does NOT contain the following files: **CONVERT/CMD**, **XFERSYS/CMD**, **LPC/CMD**, **MEMTEST/CMD**, **HERZ50/BLD**, or the one file that I consider very important: **BASIC/CMD**.

I think the documentation should have specifically explained this, as well as guiding a 'new' user step by step through the process of copying all the above mentioned protected files over to the new TRSDOS 1.4. master, either by giving the passwords, or the PATCH to disable password recognition.

THE FIX

While this isn't a problem to most experienced users, copying protected files over to the new master can present many frustrations to a novice TRSDOS user. To minimize sweat and potentional foul language, here is the step by step solution:

1. Boot up with the new TRSDOS 1.4. master disk.

2. Make the patch to disable all passwords:

PATCH SYS2/SYS (ADD = 4ED4,FIND = 20, CHG = 18) <ENTER>

3. Insert the TRSDOS 1.3. master in drive :1

4. Copy any of the files from TRSDOS 1.3. master to the TRSDOS 1.4. master. For example:

COPY BASIC/CMD:1 :0 <ENTER>

CONCLUSION

This is a very attractive package for Model III TRSDOS users with double sided drives. As a matter of fact, it may even give single sided users of TRSDOS the incentive to get the double sided drives.

PATCHER does exactly what it is supposed to do. It does it fast and it does it well. It makes working with TRSDOS a pleasure. I like it.

PATCHER is available at **$29.95** from:

DBSIDE
Suite 209 - 1051 KLO Road
Kelowna, British Columbia. Canada V1Y 4X6
(604) 762-8593

WINDOWS IN BASIC

by Robert Doerr

Why windows?
Mainly, I think, to get
the attention of the user.

There are two kinds of windows to be written in Basic. I call them informative and active windows. If the program can restore a screen, then a window consisting of an attention-getting informative message can be displayed within a border, say, upon certain data-entry errors.

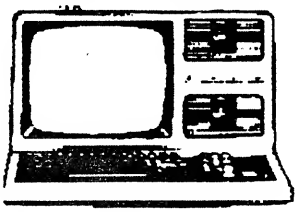
The last line of such a message would typically be: "Hit <ENTER> to continue". Upon an <ENTER>, the original screen would be restored.

An active window would be used for, say, a menu. Instead of the common "Please enter your selection by number" sort of line on a menu screen, with an active window, selection could be made by positioning the cursor at, or on, the option desired and hitting <ENTER>.

The menu could be a menu of menus; the positions of succeeding windows could serve to indicate the depth to which the selection process has gone. The number of columns of options displayed is limited only by the space on the screen; this blurs any distinction between such windows and full-screen operations.

I wrote the following small program to demonstrate the process. This is a complete program, so you may type it in and RUN it.

```
10 REM Demonstrate MENU WINDOW in BASIC

20 REM (c) 1988 Robert M. Doerr

40 PRINT CHR$(15);: 'cursor off

50 PRINT@(0,16),"╔";STRING$(25,35)"╗";:
FOR R=1 TO 21:
PRINT@(R,16),"#";SPACE$(23);"#";:
NEXT:
PRINT@(21,16),"╚";STRING$(25,35)"╝";

88 DEF FNINVVID$(A$)=CHR$(16)+A$
+CHR$(17)

90 DIM OP$(3):
FOR I=1 TO 3:
READ OP$(I):
NEXT:
DATA OPTION 1,OPTION 2,OPTION 3,"ONE  ",
"TWO  ","THREE":
FOR I=1 TO 3:
READ RESP$(I):
NEXT

100 PRINT@(2,26),"MENU";

111 FOR I=2 TO 3:
PRINT@(3+2*I,20),OP$(I);
:NEXT

122 K=1:
PRINT@(5,20),FNINVVID$(OP$(1));:
K=1

130 PRINT@(11,20),"USE VERTICAL ARROWS";:
PRINT@(13,20),"TO SELECT;";:
PRINT@(15,20),"THEN <ENTER>";

133 A$=INKEY$:
IF A$="" THEN 133

144 IF A$=CHR$(10) AND K<3
THEN PRINT@(3+2*K,20),OP$(K);:
K=K+1:
PRINT@(3+2*K,20),FNINVVID$(OP$(K));:
GOTO 133

155 IF A$=CHR$(11) AND K>1 THEN
PRINT@(3+2*K,20),OP$(K);:
K=K-1:
PRINT@(3+2*K,20),FNINVVID$(OP$(K));:
GOTO 133

166 IF A$=CHR$(13)
THEN ON K GOTO 200,300,400:
ELSE 133

200 PRINT@(18,20),"CHOICE: ";RESP$(1);:
GOTO 130

300 PRINT@(18,20),"CHOICE: ";RESP$(2);:
GOTO 130

400 PRINT@(18,20),"CHOICE: ";RESP$(3);:
GOTO 130
```



CP/M - The Alternate DOS for Model 4

by Roy Beck

Rick Donley of Ohio has asked about availability of CP/M programs suitable for use on the Mod4, and about quirks which may prevent operation of specific programs on our machines.

Under Montezuma Micro's CP/M, the Mod4 has to behave in a somewhat schizophrenic mode. This is not a large problem, and is certainly not unique to the Mod4 application. CP/M expects to operate within a rather simpleminded bailiwick, consisting of it' s major parts CCP and BDOS in a 64K (or smaller) memory map. The kernel of CP/M (if I may use that term in this case), neither knows nor cares what kind of peripherals are attached to your CPU and memory. The BIOS portion of your CP/M is tailored, in our case, to fit the Mod4. In another machine, the BIOS is tailored to fit that other machine. This all sounds elementary, I know, but I am leading up to something. The differences between machines are largely in the peripherals, including the terminal (keyboard and CRT), the mass storage devices (floppy disks, hard disks, and other devices, including tape, etc), and the I/O devices (printers, modems, etc). Further, some machines have unusual Function keys which are not present on other machines.

One of the virtues of the CP/M system is the portability of programs between machines. This is accomplished by the design and PROPER USE of the BIOS calls. So long as a particular program uses only the legal BIOS calls, it should run just fine on any CP/M machine. Conversely, any program which operates by directly accessing hardware features of a specific machine is almost certainly going to malfunction if transferred to a different brand of machine. Special Function keys, unfortunately, fall into this category.

When the question arises, "Will THISPROG.COM run on my Mod4?", the answer has to be qualified, based upon whether the program respects the BIOS and uses only the legal BIOS calls, or whether the program operates by accessing some hardware feature. In the former case, it should be OK. In the latter, it probably won't function.

Let's look a little at the schizoid personality of the Mod4 under CP/M. The terminal functions of a classic CP/M machine are normally external to the computer and are housed in a separate "black box". The earliest terminals were simply Teletype machines. As CRT and keyboard packages became available, CP/M used these. In both cases, the BIOS of CP/M did little more than shuffle terminal commands over to the terminal through a cable and accept what came back by the same path. But the Mod4 (and many other machines) does not have a separate terminal. Instead, the terminal functions are an integral part of the computer, and some part of the computer's personality must be devoted to controlling the CRT display and "listening" to the keyboard for operator input. Thus in the Mod4 the BIOS must have two personalities, one which operates in the classic CP/M fashion and a second one which takes care of the terminal housekeeping functions. Naturally this complicated life for Monte (Jesse Bob Overholt) when he designed our BIOS. Our BIOS is written to emulate an ADM3A terminal, within our hardware limitations. The ADM3A has some memory and intelligence within it, so the Mod4 BIOS has to be smart enough to perform those same functions. Thus it is obvious that our BIOS is schizoid, having to perform two separate but related tasks; handling of computer I/O and emulating the behavior of an ADM3A terminal.

It is my belief that most CP/M programs in public domain (PD) will operate satisfactorily in our Mod4's. However, I am certain there are programs around which access hardware directly in the machine they were written for, and these programs will almost certainly fail in a Mod4. I suspect this to be true of some of the more exotic KayPro programs. In fact, I suspect many disk sector editing programs (those which function like SuperUtility) access the hardware directly, and are therefore unlikely to run on a Mod4. Another class of programs which may access hardware directly are those text editors and word processors written for other machines which also have their monitor functions combined with the CPU and memory. In such cases, the author may well have directly accessed screen memory because it was easy and FAST, and yielded significant advantages to him. The resulting program will therefore only run on the machine for which it was written. Modem terminal programs are another questionable area. All you can do is try them and see if they work.

Rick has also noted in his letter to me that he has found some Osborne software and some Commodore 128 programs which will execute properly on the Mod4, but he has also run into software which will not execute properly because of the lack of certain special function keys not found on the Mod4. I should note here that Monte has allowed us to generate most control characters from the Mod4 keyboard by using combinations of keys. If you can determine what control code is expected by a program in response to a special function key, it may be possible to produce that value from the Mod4 keyboard, thus allowing the program to run

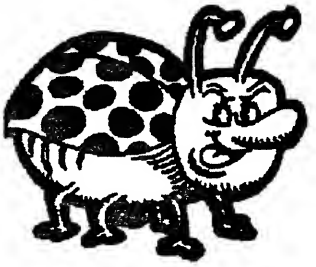
even though we don't have that particular function key. He also asked if newer or older software is more likely to run on the Mod4. Here I have to plead ignorance; sorry about that.

Rick, if you could prepare a small article on which programs run properly and which do not, and why they do not, if you know, I believe Lance would be pleased to publish it in the TRSTimes. Naturally, this would be of value to our other readers.

Incidentally, Monte (Montezuma Micro) offers an extensive library of CP/M software which will run on the Mod4. Another source of programs is Central Computer Products of Fillmore CA. They do not supply programs in the Mod4 disk format, but if you buy programs in the Kaypro 4 format, you can convert them yourself to the MM format. I personally bought Wordstar 4.0 in a Kaypro format, converted it to MM format, and am now running it on my Mod4. In conclusion on this topic, most PD CP/M software will probably run, but you can only be sure by trying it.

DDT
The CP/M DEBUG

DDT is an acronym, of course, and stands for Dynamic Debugging Tool. It has much the same kind of capabilities we expect from the DEBUG's included in the various TRS DOSes. Naturally, it has differences, and I will point out some of these.



DDT is a transient program, and must be called from disk when we need it. DDT must be called from DOS ready (A>), and may be called by itself or with the program we want to operate on. Why would we call it by itself? A couple of reasons: We might want to analyze a portion of the BIOS, or we might want to construct a small program in memory. If we are going to operate on a program, we would call DDT with the name of the program to be operated on as a parameter. An example of the latter case is as follows:

A>**DDT MYPROG.COM**

This would load both DDT and MYPROG.COM, and leave DDT in control of the machine.

What can DDT do for us? Here is a partial list:

* Fill a block of memory with a given value.
* Display memory contents in ASCII and Hex, similar to Superzap.
* Edit specific memory locations.
* Perform block moves of portions of memory.
* Enable the builtin Hex calculator for adding and subtracting Hex numbers.
* Display disassembled instructions from a region of memory (uses 8080 mnemonics).
* Assemble and insert anywhere in memory a sequence of 8080 mnemonics.
* Display and alter the CPU (8080 assumed) register contents.
* Execute code sequences, either single-stepping or with breakpoints.

Those of you who go back a few years in TRS machines may note strong similarities between DDT and RSM monitor. About the only difference is the ability of DDT to assemble and insert code into memory. Of course, RSM had some capabilities not present in DDT.

An important concern about DDT is where it resides in memory. It is different from all other CP/M programs in that it does not load into the Transient Program Area (TPA) starting at 100H. Instead, it overlays the CCP, which is not required when DDT is active, and therefore DDT does not use up part of the TPA. Thus any application program in the TPA is unaware of DDT's coresidence, and under most circumstances is unaffected by its presence. To return to normal CP/M operation, the **^C** command must be issued to cause a warm boot, which will restore CCP.

DDT has a special cursor to let you know it is present: the - or dash is used. If a program is loaded in when DDT is called, the following will appear upon completion of the load:

A>**TESTPROG.COM DDT VERS X.X**
NEXT PC mmmm nnnn

The address mmmm represents the first free address after TESTPROG.COM in the TPA, and nnnn represents the value of the program counter (PC) which DDT will use for any action it takes. This can be changed as you wish.

The command **I** with a filespec is used to create a file control block (FCB) in memory, and following this with **R** will cause DDT to load a desired program. The syntax is:

-ITESTPROG.COM

This creates the FCB and prepares for loading of the program, but does not actually load it. This will be followed by:

-R or **-R0200**

R by itself will load the program specified by **I** at the usual location in the TPA, namely 0100H. The second form of the command will allow loading anywhere in RAM. Be careful you don't shoot yourself in the foot with this one, as you could easily overlay some essential code, such as DDT, BIOS, etc.

The command **D** will display the next 192 bytes of memory in Superzap fashion, starting at the value

listed under PC as described above. If an address (in hex) is placed immediately after the **D**, then the display will begin at this address. If a comma and a second address are appended to the first address, then a range of locations will be displayed. If the range of locations is greater than 192, then **^S** must be used to start/stop scrolling.

A single value can be used to fill a block of memory. the syntax is:

-Fmmmm,nnnn,vv

where mmmm and nnnn are the limits of the block to be filled, and vv is the hex value to be used to fill the block.

The code starting at a location can be disassembled and displayed with the **L**(list) command. The syntax is:

-Lmmmm,nnnn

Unfortunately, the mnemonics will be Intel's 8080 values, which take a little getting used to. If Z-80 code is disassembled many instructions will be displayed as ?? = , which means DDT could not interpret the instruction as an Intel mnemonic. This can also be caused by attempting to disassemble data. It should be noted that there are other versions of DDT in the marketplace which can correctly handle Z-80 code. However, since all of us received Digital Research's DDT with our MM CP/M, I am explaining it even though it is not the one best suited for our purposes.

The Set command (**S**) is similar to M of the TRS Debuggers, and displays and changes memory contents. Syntax is as follows:

-Smmmm

Memory contents starting at mmmm are displayed, a byte at a time. After each byte is displayed, you have the option of changing it or going on to the next one. To leave it unchanged, just hit Enter; to change it, enter the new hex value and then hit Enter. A period "." is used to terminate the function.

DDT includes a mini-assembler which is a convenient way to patch code in memory (if you know Intel mnemonics). **A** stands for assemble. The syntax is:

-Ammmm

where mmmm is the starting address. The user types in the Intel mnemonic and any required hex values at each address, followed by Enter. The mini-assembler will insert the appropriate values into the memory locations (1, 2, or 3 locations as required). The function is terminated by Enter alone, or by a period. An erroneous mnemonic will cause a ? to be displayed and also terminate the function.

The Block move function is commanded by **M**, with syntax as follows:

-Mmmmm,nnnn,dddd

where mmmm and nnnn are starting and ending addresses of the area to be moved and dddd is the start of the destination area.

The Hex calculator function is commanded as follows:

-Hmmmm,nnnn

This command will return the sum and difference of the two hex values, mmmm and nnnn. It is frequently useful in figuring offsets or displacements in hex values. This is the same as the B command in RSM.

There are several other DDT commands, but these are primarily useful to machine language programmers, and I have already run over my alloted space for the month. Accordingly I will only mention the commands, which are:

- **G** goto and execute, with or without breakpoints
- **T** traces and displays a finite number of instructions
- **U** executes a finite number of instructions without displaying them
- **X** displays and allows alteration of CPU registers (ignores the Z-80 registers not present in an 8080 chip).

HUNTING FOR BURIED TREASURE

Peeking & Poking Model 4

by Tim Sewell & Lance Wolstrup

The Model 4 runs almost twice as fast as the Model III. Yet, a Model 4 program which makes frequent use of text or graphics on the screen will appear to run much slower than the exact same program written for a Model III.

The reason: The Model III screen is memory mapped. That is, a portion of RAM has been set aside to exclusively hold the contents of the screen. The screen begins at memory location 3C00H (15360 in decimal) and goes on for the next 1023 memory locations. The Model 4 screen is NOT memory mapped. Instead of having an actual physical screen reside at a particular series of memory locations, the authors of TRSDOS 6.x and LS-DOS 6.x opted for a different way of accessing the screen.

This method gives us a few extra 'K' of RAM, but there are, unfortunately, a couple of trade offs. Screen access is very slow, and direct communication with the screen is not possible:

You CAN'T PEEK and POKE the screen.

(Boo, hiss, loud cat-calls, etc.)

OH YEAH??

While liking the extra 'K' of RAM, I certainly would much prefer to have a screen that would be speedy and could be manipulated with the PEEK and POKE commands. After talking to other members of the Hackers' Group over uncountable cups of coffee, then disassembling and studying strategic portions of DOS, I finally came to the conclusion that the 'impossible' is indeed **POSSIBLE**. We can speed up the screen and, better yet, we can use the PEEK and POKE commands to communicate with it directly.

The Model 4 Technical Reference Manual states that **'Video RAM exists from F800H to FFFFH'**.

This is one of those famous half truths that confused us all for a while. The real truth is that this memory is only Video RAM **some** of the time. It is shared memory. Most of the time it is **NOT** Video RAM. That is why you cannot PEEK and POKE it.

A clever DOS routine starting at 817H determines whether the video should be enabled or disabled. While this routine does many important housekeeping chores, I will only discuss its more elementary points, the ones important to our task.

Memory location 78H (OPREG$, in the flag table) holds information about the screen. This location is called the 'PORT 84H image'. PORT 84H controls the video. Whenever we want to do something special to the screen, we put the appropriate value into 78H, and then **OUT 84H** with the value from location 78H. (We did this in issue 4 to get the 80 column screen from Model III mode.)

To enable the Video RAM, the DOS routine first picks up the value from 78H. It then proceeds to **AND** this value with **FCH**. Since the bits of FCH are **1111 1100**, it becomes obvious that bit 0 and 1 are being forced off, while the the rest of the bits are preserved. Then the value is **OR**'ed with **82H (1000 0010)**. In other words, bit 7 and bit 1 are mandatorily turned on while bit 0 remains off. This value is now sent to PORT 84H. Video RAM is enabled and thus, the screen is updated.

The routine then immediately proceeds to disable Video RAM. This is done by OUT 84H with the **ORIGINAL** value from 78H.

Confusing? - You bet it is!!

To make some sense out of this, get into Basic. Remember that Video RAM is always, by default, disabled. Let's find out what the value of 78H is:

PRINT PEEK(&H78) <ENTER>

You should see the decimal value **135**, which translates to **87H**, or **1000 1111B**. Note that bit 0 is turned on. From this we can conclude that:

When Video RAM is disabled, bit 0 of 78H is **ON**.

Contrast this to the above explanation of enabling Video RAM. There bit 0 remained **OFF**.

Conclusion: Turning **OFF** bit 0 of value in 78H enables Video RAM.

Let's test this theory. First, turning off bit 0 of the value 135 changes it to 134. (Not too difficult!)

Now type:

POKE &H78,134:OUT &H84,134 <ENTER>

What happened? Most likely, your machine froze up (CRASH).

Well, don't just sit there! Go ahead, REBOOT.

Does that mean that our theory is wrong?

No, DOS just got temporarily confused. We can rectify this by setting HIGH$ to the start of Video RAM. This could be done from DOS, before we go into Basic, by setting high memory to F7FFH:

BASIC (M=63487) <ENTER>

Setting the top of high memory from DOS is inconvenient. I prefer to do it while in Basic. Here we have a command that will do it for us: **CLEAR**.

This command has changed somewhat from Model III Basic. It is now capable of setting the highest memory location which Basic can access. Get in Basic and type this:

CLEAR,&HF7FF <ENTER>
POKE &H78,134:OUT &H84,134 <ENTER>
This time you didn't crash. Instead you have just enabled Video RAM permanently. You can now PEEK and POKE the screen. Try it by typing:
POKE &HF800,191 <ENTER>

How about that! The pixel at 0,0 lit up. Now try to read the pixel:
PRINT PEEK(&HF800) <ENTER>

The value **191** is displayed on the screen. You just PEEKed and POKEd the screen. It starts at &HF800 and continues for the next 1919 memory locations, AND you can access any and all directly.

Since the screen is now permanent, regular screen commands will be executed faster. The trade-off, you just lost 2K of user RAM. You can, however, restore the startup values by:
CLEAR,&HFFFF <ENTER>
POKE &H78,135:OUT &H84,135 <ENTER>

With this information, new and interesting programs can be written. Model I and III programs which use direct screen access can be translated to Model 4 with ease. All kinds of possibilities are now available.

LABEL 204

A few months ago, Tim Sewell converted an old Model III public domain program to run on the Model 4. This program, called LABEL200, creates disk labels on the DMP200 printer. As it is a very useful program, we decided to modify the translation to demonstrate just how fast direct screen access can be.

Well, one thing led to another and before long, Tim and I had modified the modified modifications to the point that a brand-new LABEL200 was born. This one, because it is strictly for Model 4, is called LABEL204/BAS.

The original program consisted of a series of menus to guide you through the creation of a disk label. When you had selected the appropriate categories, a neat label would then be printed. This was written specifically for the DMP200.

The new version sticks to the original concept. It also uses menus for the user to choose the various information for a disk label. But no longer do the menus scroll slowly onto the screen. They now appear instantly - as WINDOWS.

To explain the program, let's look at each window in turn. As the program is loading and going through some important set-up routines, a minor front screen is displayed. Then, instantly, the first window pops to the screen. At the top you'll see the program name and the credits. Below is the menu for the label header, the one decribing the program type. There are 20 choices, ranging from OPERATING SYSTEM, EDUCATION, WORD PROCESSING to ARCADE GAMES, COMMUNICATIONS or DATA BASE.

Should you find that none of the listed choices fits your disk, you have an option where you can type your own header. You can also exit to DOS from this window. Note that selections are not made by typing the usual number or letter; instead you move a pointing hand next to the desired selection and press <ENTER>. The hand is moved by pressing the arrow keys. Full wrap-around in all directions is employed.

When <ENTER> is pressed, the category being pointed to by the hand is selected, and the next window is displayed.

Be aware that, except for the first window, you may step back to the previous window by pressing <F1>.

Window 2 simply allows you to type the name of the disk - it's your preference.

Window 3 enables you to select the DOS type.

Window 4 is used for selecting the number of tracks and the format density of the disk.

Window 5 prompts you to type additional information about the disk - again, your choice.

At this point the label has been created and is displayed in a window towards the top of the screen. Observe that though you were not prompted for the date, it appears on the right hand side of the third line of the label display. If your Model 4 is not using a date then it is displayed as 00/00/00.

You now have five choices:

PRINT LABEL - choosing this option prompts you for the number of labels you wish to print. 999 is the maximum. After choosing the number of labels, you are prompted to type a starting serial number. If a number is typed, then each label will carry an incrementing serial number. If a serial number is not desired, simply press <ENTER> and your labels will be printed.

EDIT LABEL - When perusing the label display toward the top of the screen, you may find that you wish to change one or more of the entries. Choose EDIT LABEL and press <ENTER>. Now point the hand to the entry you wish to change and press <ENTER>. You will now be able to type your correction. You may continue to edit any or all of the entries. When done, press <F1>.

ALIGN LABEL - this option help you position your labels correctly in the printer. It will fill one label with asterisks.

SELECT PRINTER - Whereas the original program only supported the DMP200, we have included the printer codes for Epson and compatible printers.

REBOOT DOS - You'll be prompted to confirm this choice by typing Y. Doing so leaves Basic and completely reboots the system. Any other key brings back the window you were in.

Note that the <BREAK> key is disabled during program execution. If you absolutely INSIST on

breaking out of the program, we have included a 'back-door' command. Pressing <**CLEAR**> <**B**> enables the <BREAK> key.

While program is for the most part self-explanetory to use, the programming is anything but that, so let's talk briefly about the most interesting portions of the code.

LABEL204/BAS is written in 99% pure Basic. (YES, I admit it - we did use two teenie-weenie machine language routines - I hang my head in shame).

The two ML routines were used to obtain the necessary speed to 'push' the window onto the screen as whole, rather than scrolling it visibly to the naked eye. They are both simple LDIR routines and they perform a variety of duties.

The program begins at line **1000**. Here, among other things, we protect all memory from C000H to the top. The number of printer drivers are set to 2 (PD = 2) and the default printer is set to 1 (PDF = 1).

Line **1005** disables the <BREAK> key, enables the 'constant' screen, forces special characters, and finally, turns off the cursor.

Line **1010** reads the first machine language routine from line 20 and POKEs it into protected memory at FFF0H. At this point it is a 'dummy' routine. It reads as follows:

```
LD    HL,0000H
PUSH  HL
POP   DE
INC   DE
LD    BC,0000H
LD    (HL),0
LDIR
RET
```

Line **1020** reads the second ML routine (from line 30) and POKEs it into protected memory at FFE0H. This routine is also a 'dummy' at this point. It reads:

```
LD    HL,0000H
LD    DE,0000H
LD    BC,0000H
LDIR
RET
```

The address of the first ML routine is stored in variable ML1. The second routine's address is stored in ML2.

So far, so good, but you might be wondering just why we have bothered to POKE in two ML routines that are unusable. The answer is, we have established two ML routines that can be PROGRAMMED from within Basic. We will use this concept in line **1100**. Here we set up the following variables: H = &HC0:L = 0:B = 7:C = &HFF:CH = 32: and then we proceed to the subroutine in line **100** to program the first ML routine. What is happening is that the value of H is POKEd into ML1 + 1, which is FFF1H. This is followed by POKEing the value of L into ML1 + 2 (FFF2H). Then the value of C is POKEd into ML1 + 7, followed by POKEing the value of C into ML1 + 8. Lastly, the value of CH is POKE into ML1 + 10.

The machine language routine starting at FFF0H has now been modified to read:

```
LD    HL,C000H
PUSH  HL
POP   DE
INC   DE
LD    BC,07FFH
LDIR
RET
```

By CALLing ML1 in line **110** we execute the machine languge routine. In this particular case the machine language routine erased the 2048 consecutive memory locations starting at C000H (sort of a non-visual CLS).

It could have been written in straight Basic as:

```
FOR X=&HC000 TO &HC001+2047:
POKE X,32:NEXT
```

This would have been unacceptably slow, so we opted to do it with the much, much faster machine level code.

Line **1110** sets D = &HC2:E = &HEF:A$ = "L A B E L 2 0 4" and proceed to the subroutine in line **200**. This is the routine that employs the second ML routine. You might think of it as a very fast PRINT@ command. It immediately sets B = 0. Then, by **PEEK(VARPTR(A$))**, it finds the length of A$ and puts that in variable C. Next it puts the Least Significant Byte (LSB) of the address of A$ into L, followed by stuffingt the Most Significant Byte (MSB) of the address of A$ into H.

Line **210** POKEs the value of L into ML2 + 1, the value of H into ML2 + 2. The value of E is POKEd into ML2 + 4, D into ML2 + 5, C into ML2 + 7 and B into ML2 + 8. If you have ever attempted anything at all in Assembly Language, you will notice that we are actually loading the HL, DE and BC registers directly from Basic.

The second machine language routine is now modified to this:

```
LD    HL,address of A$
LD    DE,0C2EFH
LD    BC,length of A$
LDIR
RET
```

Line **220** CALLs ML2 to execute the routine. The text in A$ has been stored beginning at memory location C2EFH. This could have been written in straight Basic as:

```
FOR X=0 TO LEN(A$)-1:
POKE &HC2EF+X,MID$(A$,X+1,1):
NEXT
```

Line **1120** does the same as line 1110, but with a different value in A$ and another PRINT@ memory location in D and E.

Lines **1140** to **1160** POKEs a border around the text.

All of the above code has produced a temporary screen in memory. It is NOT visible at this point. Remember that we have established the memory starting at F800H as video memory. Since whatever we put there will be displayed on the CRT, we will copy the 1920 memory locations (80X24) starting at

C000H over to the video memory block starting at F800H.

Line **1180** puts the values in H and L to indicate the start of the screen to be copied. Then we GOSUB **900** where D and E are given values that point to the start of video memory. In line **910** we give B and C the value of how many bytes to copy: 0780H (1920). Then we GOTO **210** where, just as before, we actually load registers HL, DE and BC with the above values. Line 220 executes the machine language routine and the temporary screen at C000H is copied onto the visual screen at F800H. **BAM - FAST!!**

While this screen is being viewed, the windows are being created in memory using the exact same routines as above. (lines **1190** to **1910**).

The code beginning at line **2000** starts the actual visual program. After copying the the menu choices into the X$ array, setting the appropriate values to move the pointing hand correctly, the first menu screen is thrown onto the CRT in line **2010**.

There are plenty other techniques of interest, but I am running out of room, so should you have questions about a particular sequence of code, be sure to write and let me know. I will then try to cover the questions in detail in upcoming issues. I will close by pointing out that ending the program causes a complete reboot. This is done in line **3610** where **A=&H1BF2:CALL A** is a call to the @IPL SVC which does a software reset. Why reset the machine? Because the program consumes almost all the available memory. Simply ENDing would leave you with anywhere from 400 bytes to 1K of memory. Typing NEW leaves 13540 bytes, so my preference is to reboot and thus restore all values.

And now ·····> **heeeeeere's TIM!**

The Full (But Possibly Not Complete) History of LABEL204/BAS

by Timothy Sewell

I had been searching for a decent label printing program ever since I bought my first Model 4P back in 1984 (seems like only yesterday). Sure, there were plenty of label generating programs to be found in the Public Domain, but it seemed like ALL of them were written to be used on an EPSON printer with advanced graphic capabilities. Since I own a DMP 200 printer, the programs would not work, nor could they be converted since the DMP 200 doesn't have Super-Micro-Squashed-Smooshed-Whatever type of compression. It's just a faithful ol' workhorse that hasn't let me down yet!

So, what was I to do? I had bought this computer to learn about programming, but I was bitten by the BBS bug and spent all of my computer time running up phone bills that would rival the national debt. That meant writing one on my own was definitely out. To my knowledge there were no commercial programs available that supported the DMP codes, so I was out of luck there. I eventually resorted to using the form letter and type expansion features of LeScript (my favorite word processor) to create my labels. This worked just fine for my needs and I went on quite content with the knowledge that my diskettes were now properly labeled.

In the summer of 1986, I had just called one of my favorite BBS systems up in Northern California and got a suprising banner as the opening message. The Marin-80 BBS had been operating for several years on a Model 4 under the TBBS bulletin board program (the best ever written for the Model III). TBBS was now going to drop support of the TRS-80 version and release an new enhanced version for the IBM/MS-DOS community. Will Gortner, the Sysop of Marin-80, had just bought an IBM clone and the new version of TBBS, so he wanted to sell his complete Model 4 system which included all of his software and hard drive.

The price for the system was right so I called Will and arranged a weekend where I could drive up to the San Francisco area and pick it up. After a pleasant 16 hour drive in my (then) new car, I was back home to fully explore the spoils of my purchase.

I had noticed the nice format of the labels that were on the disks. I had also noticed that Will owned a DMP 200 (which he kept for his new BBS). After looking over several of the disks I found a program written in Model III BASIC called LABEL/BAS. I tried out the program and I was pleased to find that it produced labels the same way as the labels on the disks I just obtained. I was in heaven!

The code was very simple and to the point. You typed in the information you wanted and the label came out on the printer. After a closer look at the code, it appeared that someone was attempting to convert the program to Model 4 BASIC, but had given up. I just left it alone and continued to make my labels. This worked just fine for a few months until my disk collection started to grow so large that I needed to speed up my label-making process.

LABEL/BAS was quite tedious on a long haul due to the fact that you had to physically type EVERY bit of information for EACH label you wanted to print. That came out to a lot of typing and I thought I had bought this computer to SAVE me time and work (HA!).

So, I finally sat down, un-plugged the phone line, and actually started to modify the code "borrowing" routines from the program and asking a few questions in the BBS community. After a few days I came up with a program that did a pretty good job of what I wanted. Names and titles that I used often were now available to me via a menu and single keystroke, as well as the type of DOS and disk format information. The automatic date feature from the original program was retained and I was quite pleased

with the end results. As a matter of fact, so pleased that I decided to release it to the public domain. There was one problem (at least to me). The program was ugly! It had no life to it... no graphics.

Graphics. That was a dreaded word to me. Hey, I just banged my non-programmers brain for several days writing code and now I expected myself to add graphics to this thing? Maybe I should forget about releasing it to the public domain and save myself the trouble of having to learn something more. I was already developing the "shakes" from not being on the phone lines for the last couple of days. I knew withdrawal would be setting in soon, but I thought about all those other DMP printer owners who would probably love me to death for finally making a program available that works on their printers. My ego kicked in. I was going to do it even if it killed me.

I started looking at other BASIC programs that used graphics and found that there were two ways of adding graphics to a program. You could write code (ugh!) that added the graphics or you could actually draw the graphics and incorperate it into the BASIC code. I naturally opted for drawing.

The next challenge was finding a program that could EASILY handle drawing graphics into an existing file. My answer came in a little known program called GRAPHICS-90. Graphics-90 was written by Larry Payne of Software Affair (The Orchestra-90 people) and was to originally be distributed by Radio Shack. The Shack dragged their feet on getting the ball rolling for Graphics-90 and the author eventually withdrew the submission and allowed the program to be distributed through some of the large information networks such as DELPHI. This, unfortunately, resulted in a beautiful program being completely overlooked by the majority of TRS-80 owners. If you have ever played the game 13 GHOSTS, the graphic and animation sequences were written completely with Graphics-90.

Graphics-90 has the feature of allowing you to take an existing ASCII file and adding graphics to it. This is perfect for adding graphics to a BASIC file saved in ASCII. The one minor hurdle was that Graphics-90 will ONLY work under TRSDOS 1.3., and LABEL/BAS was written under LDOS. This was easily rectified by the use of Super Utility 4/4P. Once the file was transfered to a TRSDOS 1.3 format disk, I was ready to add my graphics. After a couple of hours of "learning" how to use Graphics-90 while adding my graphics to the program, I ended up with what I thought was a pretty sharp looking program. I dumped the graphics to my printer for reference and away I went. I should note here that Graphics-90 will ONLY dump to a DMP series printer. No other printers are supported (He! He!).

I decided that this program was a far cry from the original LABEL/BAS so a new name was in order. The program was thus dubbed LABEL200/BAS in honor of the printer for which it was originally written. I released it into the public domain and many thank you's were bestowed upon me from various BBS systems. There was FINALLY a label program exclusively for the DMP printers.

This worked just fine for me except (there's that dreaded word again) that I was becoming more and more involved with the Model 4's native mode and it was kind of a pain to have to re-boot under LDOS every time I wanted to make some labels. So I accepted the personal challenge of converting LABEL200/BAS to run on the Model 4. This took several weeks of searching for alternatives to the Model III code. Two great sources of information that I used were "Learning TRS-80 Model 4/4P BASIC" by David A. Lien and "BASIC Program Conversions" by The Editors of Computer Skill Builders.

After hacking around and once again using Graphics-90, I came up with a working Model 4 version of LABEL200/BAS and was quite proud of myself. I showed the program to Lance who was looking for a vehicle to demonstrate the powerful abilities of his PEEKing and POKEing routines in Model 4 BASIC and he immediately asked me if he could "play" with LABEL200 a bit. Now "play" is a very strange word where Lance is concerned. When he "plays" with something you better resolve yourself to seeing a whole new generation of the program you started with. LABEL200 was indeed crude. There were several features I would have liked to add to it but couldn't due to lack of programming knowledge. I discussed these enhancements with Lance and he just looked at me with this "ALF" like grin and said "No Problem". I should have known.

So what you have before you literally took years to create. It is a far cry from the original LABEL/BAS found many years ago. This new version gives you the ability to create and edit your label. The free form entry ability means that you are not limited to the one menu for your heading choices (I used to keep anout 3 different version of the older program for all of the different headings I would use). The window abilities are quite staggering where the Model 4 BASIC is concerned. And to make the program more universally accepted, you can use it with an EPSON printer as well (even though the default remains the DMP 200).

So, kick back, enjoy, and most of all, LEARN from this program. There are some features that have never been seen before. You may wish to one day incorperate some of the "tricks" that are used into your own programs, or maybe show us a few "tricks" of your own.

I have learned a lot from LABEL204/BAS. I hope you will too.

LABEL204/BAS

```
10 GOTO 1000
20 DATA 33,0,0,229,209,19,1,0,0,54,0,237,
176,201
30 DATA 33,0,0,17,0,0,1,0,0,237,176,201
40 DATA OPERATING SYSTEM,UTILITIES,BUSIN
ESS,COMMUNICATIONS,WORD PROCESSING,P
UBLIC DOMAIN,ENTERTAINMENT,DATA BASE,E
DUCATION,HOUSEHOLD
45 DATA TRS-80 LINK,TRS-80 MODEL 4,TRS-80
MODEL I/III,HIGH RESOLUTION,CP/M,THE FILE C
ABINET,ARCADE GAMES,THE ARRANGER,TYPE
YOUR OWN,REBOOT DOS
50 DATA SELF BOOTING,TRSDOS 1.3.,TRSDOS
6.2.x.,LS-DOS 6.3.,LDOS 5.1.x.,NEWDOS/80,MO
DEL I FORMAT
55 DATA DOSPLUS 3.5.,DOSPLUS 4,MULTIDOS
1.7.,MULTIDOS 80/64,MONTEZUMA CP/M,CP/M
PLUS 3.0.,DATA DISK
60 DATA 35 Trk - SS,40 Trk - SS,40 Trk - DS,80
Trk - DS,Self Booting Format
70 DATA PRINT LABEL,EDIT LABEL,ALIGN LABE
L,SELECT PRINTER,REBOOT DOS
75 ' DMP & Epson printer codes
80 DATA 27,14,27,15,27,19,27,20,27,23,27,31
85 DATA 27,14,27,20,27,64,27,15,27,15,27,69
90 DATA DMP-200,EPSON FX-80
95 ' poke values for LDIR routine
100 POKE ML1 + 1,L:POKE ML1 + 2,H:POKE
ML1 + 7,C:POKE ML1 + 8,B:POKE ML1 + 10,CH
110 CALL ML1:RETURN
195 ' Poke values for LDIR routine to display
196 ' text and move screens back and forth
200 B = 0:C = PEEK(VARPTR(A$)):L = PEEK
(VARPTR(A$) + 1):H = PEEK(VARPTR(A$) + 2)
210 POKE ML2 + 1,L:POKE ML2 + 2,H:POKE ML2
+ 4,E:POKE ML2 + 5,D:POKE ML2 + 7,C:POKE
ML2 + 8,B
220 CALL ML2:RETURN
295 ' INKEY$ subroutine
300 PRINT@FC,STRING$(MC,46);: PRINT@FC,
"";: POKE &HB97,32:A$ = "":CC = 0:CL = FC
310 I$ = INKEY$:IF I$ = "" THEN 310
320 IF I$ = CHR$(13) OR I$ = CHR$(129) THEN
POKE &HB97,0:RETURN
330 IF I$ = CHR$(8) AND CL = FC THEN 310
331 IF I$ = CHR$(9) OR I$ = CHR$(10) OR
I$ = CHR$(11) THEN 310
340 IF I$ = CHR$(8) THEN PRINT@CL,ED$;:
CL = CL-1:CC = CC-1:A$ = LEFT$(A$,CC):
GOTO 310
350 IF MC = CC THEN 310
360 PRINT@CL,I$;:A$ = A$ + I$:CL = CL + 1:
CC = CC + 1:GOTO 310
400 PL(1) = 260 + INT((41-LEN(LB$(1)))/2):
PL(2) = 340 + INT((41-LEN(LB$(2)))/2):
PL(3) = 415:PL(4) = 420 + INT((41-LEN(LB$(4)))/2):
PL(5) = 458:PL(6) = 500 + INT((41-LEN(LB$(6)))/2):
RETURN
495 ' Routine to move pointer
500 PP2 = PP1 + LEN(A$):PO = P1:PRINT@P1,
PT$;:PRINT@PP1,A$;:PRINT@PP2,X$(P2);
510 I$ = INKEY$:IF I$ = "" THEN 510 ELSE IF I$ =
CHR$(194) THEN POKE &H7C,PEEK(&H7C)
AND 239
515 IF I$ = CHR$(129) AND S = 1 THEN 510
517 IF I$ = CHR$(13) OR I$ = CHR$(129) THEN
FL = 0:S = 0:RETURN
520 IF I$ = CHR$(10) THEN PRINT@P1,ER$;:P1 =
P1 + 80:P2 = P2 + 1:IF P1 = CL1 THEN P1 = PO +
NC:CO = 2:IF FL THEN P2 = 1:GOTO 590
530 IF I$ = CHR$(10) AND P1 = CL2 THEN
P1 = PO:CO = 1:P2 = 1
540 IF I$ = CHR$(11) THEN PRINT@P1,ER$;:
P1 = P1-80:P2 = P2-1:IF P1 = PO-80 THEN
P1 = CL2-80:CO = 2:P2 = SI
550 IF I$ = CHR$(11) AND P1 = PO + NC-80
THEN P1 = CL1-80:CO = 1
560 IF FL THEN 590 ELSE IF I$ = CHR$(9) THEN
PRINT@P1,ER$;:IF CO = 1 THEN P1 = P1 + NC:
CO = 2:P2 = P2 + SI/2 ELSE IF CO = 2 THEN
P1 = P1-NC:CO = 1:P2 = P2-SI/2
570 IF I$ = CHR$(8) THEN PRINT@P1,ER$;:IF CO
= 1 THEN P1 = P1 + NC:CO = 2:P2 = P2 + SI/2 ELS
E IF CO = 2 THEN P1 = P1-NC:CO = 1:P2 = P2-SI/2
590 PRINT@P1,PT$;:PRINT@PP2,ER2$;:
PRINT@PP2,X$(P2);:GOTO 510
895 'Routine to copy & switch screens
900 D = &HF8:E = 0: ' Choose visual screen
910 B = 7:C = &H80:GOTO 210: 'counter is 1920
995 ' Program begins here
1000 DEFINT A-C,X:CLEAR,&HBFFF:OPTION BAS
E 1:PD = 2:PDF = 1:DIM B$(20),C$(14),D$(5),E$(5)
,X$(20),LB$(7),PL(6),LE(6),PRT$(PD,6),PRT(PD)
1002 PT$ = CHR$(143) + CHR$(244) + CHR$(245)
+ CHR$(246):ER$ = STRING$(4,32):
ER1$ = STRING$(20,32):ER2$ = STRING$(45,32):
ED$ = CHR$(8) + CHR$(46) + CHR$(24)
1005 POKE &H7C,PEEK(&H7C) OR 16:POKE
&H78,PEEK(&H78) AND 254:OUT &H84,PEEK
(&H78): POKE &HB94,PEEK(&HB94) OR 8:
POKE &HB97,0
1010 ML1 = &HFFF0:FOR X = 0 TO 13:READ A:
POKE ML1 + X,A:NEXT
1015 ' Poke LDIR routine for text into FFE0H
1020 ML2 = &HFFE0:FOR X = 0 TO 11:READ A:
POKE ML2 + X,A:NEXT
1095 ' Erase screen 1 and set up front screen
1100 H = &HC0:L = 0:B = 7:C = &HFF:CH = 32:
GOSUB 100
1110 D = &HC2:E = &HEF:A$ = "L A B E L  2 0 4":
GOSUB 200
1120 D = &HC3:E = &H84:A$ = "A fancy disk label
program for Model 4":GOSUB 200
1140 LO = &HC292:POKE LO,151:
POKE LO + 41,171:H = &HC2:L = &H93:B = 0:
C = 39:CH = 131:GOSUB 100
1150 FOR X = 1 TO 4:POKE LO + X*80,149:
POKE LO + X*80 + 41,170:NEXT
1160 LO = &HC422:POKE LO,141:
POKE LO + 41,142:H = &HC4:L = &H23:B = 0:
```

```
C = 39:CH = 140:GOSUB 100
1180 H = &HC0:L = 0:GOSUB 900
1190 H = &HC0:L = 0:B = 7:C = &HFF:CH = 32:
GOSUB 100
1195 ' Create screen 1 at C000H
1200 LO = &HC000:POKE LO,151:
POKE LO + 79,171:H = &HC0:L = 1:B = 0:C = 77:
CH = 131:GOSUB 100
1210 FOR X = 1 TO 3:POKE LO + X*80,149:
POKE LO + X*80 + 79,170:NEXT
1220 POKE LO + X*80,141:
POKE LO + X*80 + 79,142: H = &HC1:L = &H41:
B = 0:C = 77:CH = 140:GOSUB 100
1230 D = &HC0:E = &H52:A$ = "TRSTimes
Presents: " + PT$:GOSUB 200:E = &H70:
A$ = "L A B E L  2 0 4":GOSUB 200:E = &H87:
A$ = "Original author unknown":GOSUB 200
1240 E = &HAD:A$ = "Modifications and transla
tion to Model 4 by Tim Sewell":GOSUB 200:
D = &HC1:E = &H0:A$ = "Machine code support
routines by Lance Wolstrup":GOSUB 200
1250 LO = &HC1E5:POKE LO,151:
POKE LO + 69,171:H = &HC1:L = &HE6:B = 0:
C = 67:CH = 131:GOSUB 100
1260 FOR X = 1 TO 10:POKE LO + X*80,149:
POKE LO + X*80 + 34,170:POKE LO + X*80 +
35,149:POKE LO + X*80 + 69,170:NEXT:POKE
LO + 80*X,141:POKE LO + X*80 + 69,142:
H = &HC5:L = &H56:B = 0:C = 67:CH = 140:
GOSUB 100
1270 LO = &HC23E + 65536!
1280 FOR X = 0 TO 9:READ B$(X + 1):
A$ = B$(X + 1):D = INT((LO + X*80)/256):
E = LO + X*80-(D*256):GOSUB 200:NEXT
1290 FOR X = 0 TO 9:READ B$(X + 11):
A$ = B$(X + 11):D = INT((LO + X*80 + 35)/256):
E = LO + X*80 + 35-(D*256):GOSUB 200:NEXT
1300 H = &HC6:L = &HE0:B = 0:C = 79:GOSUB
100:D = &HC7:E = &H42:A$ = "Use Arrow keys to
select - then press <ENTER> ":GOSUB 200
1395 ' Copy screens for window effect
1400 H = &HC0:L = 0:D = &HC7:E = &H80:
GOSUB 910: ' Copy to screen 1 to screen 2
1410 LO = &HC82E + 65536!:POKE LO,151:
POKE LO + 52,171:H = &HC8:L = &H2F:B = 0:
C = 50:CH = 131:GOSUB 100
1420 A$ = CHR$(149) + STRING$(51,32) +
CHR$(170): FOR X = 1 TO 4:
D = INT((LO + X*80)/256):E = LO + X*80-(D*256):
GOSUB 200:NEXT
1430 LO = &HC9BE:POKE LO,141:
POKE LO + 52,142:H = &HC9:L = &HBF:B = 0:
C = 50:CH = 140:GOSUB 100
1440 LO = &HCAA9:POKE LO,151:
POKE LO + 61,171:H = &HCA:L = &HAA:B = 0:
C = 59:CH = 131:GOSUB 100:H = &HC7:L = &H80:
D = &HCF:E = 0:GOSUB 910: ' Copy 2 to 3
1450 D = &HCA:E = &HF9:A$ = CHR$(149) +
STRING$(60,32) + CHR$(170):GOSUB 200
1460 LO = &HCB49:POKE LO,141:
POKE LO + 61,142:H = &HCB:L = &H4A:B = 0:
C = 59:CH = 140:GOSUB 100
1500 LO = &HD229 + 65536!:A$ = CHR$(149) +
STRING$(29,32) + CHR$(170) + CHR$(149) +
STRING$(29,32) + CHR$(170):FOR X = 1 TO 7:
D = INT((LO + X*80)/256):E = LO + X*80-(D*256):
GOSUB 200:NEXT
1510 LO = &HD4A9:POKE LO,141:
POKE LO + 61,142:H = &HD4:L = &HAA:B = 0:
C = 59:CH = 140:GOSUB 100
1520 LO = &HD233 + 65536!:FOR X = 1 TO 7:
READ C$(X):A$ = C$(X):D = INT((LO + X*80)/256):
E = LO + X*80-(D*256):GOSUB 200:NEXT
1530 LO = &HD233 + 31 + 65536!:FOR X = 1 TO 7:
READ C$(X + 7):A$ = C$(X + 7):
D = INT((LO + X*80)/256):E = LO + X*80-(D*256):
GOSUB 200:NEXT
1600 H = &HCF:L = 0:D = &HD6:E = &H80:
GOSUB 910: ' Copy 3 to 4
1610 LO = &HDA09 + 65536!:POKE LO,151:
POKE + LO + 30,171:H = &HDA:L = &HA:B = 0:
C = 28:CH = 131:GOSUB 100
1620 A$ = CHR$(149) + STRING$(29,32) +
CHR$(170):FOR X = 1 TO 5:
D = INT((LO + X*80)/256):E = LO + X*80-(D*256):
GOSUB 200:NEXT
1630 LO = &HDBE9 + 65536!:POKE LO,141:
POKE LO + 30,142:H = &HDB:L = &HEA:B = 0:
C = 28:CH = 140:GOSUB 100
1640 LO = &HDA09 + 9 + 65536!:FOR X = 1 TO 5:
READ D$(X):A$ = D$(X):D = INT((LO + X*80)/256):
E = LO + X*80-(D*256):GOSUB 200:NEXT
1700 H = &HD6:L = &H80:D = &HDE:E = 0:
GOSUB 910: ' Copy 4 to 5
1710 LO = &HE126 + 65536!:POKE LO,151:
POKE LO + 66,171:H = &HE1:L = &H27:B = 0:
C = 64:CH = 131:GOSUB 100
1720 A$ = CHR$(149) + STRING$(65,32) +
CHR$(170):D = INT((LO + 80)/256):
E = LO + 80-(D*256):GOSUB 200
1730 LO = &HE1C6 + 65536!:POKE LO,141:
POKE LO + 66,142:H = &HE1:L = &HC7:B = 0:
C = 64:CH = 140:GOSUB 100
1740 A$ = "Additional Information: " + STRING
$(38,46):LO = &HE128 + 65536!:D = INT((LO + 80)
/256):E = LO + 80-(D*256):GOSUB 200
1800 H = &HDE:L = 0:D = &HE5:E = &H80:
GOSUB 910: ' Copy 5 to 6
1810 LO = &HE850 + 65536!:POKE LO,151:
POKE LO + 79,171:H = &HE8:L = &H51:B = 0:
C = 77:CH = 131:GOSUB 100
1820 A$ = CHR$(149) + STRING$(21,32) +
CHR$(170) + STRING$(56,32) + CHR$(170):
FOR X = 1 TO 10:D = INT((LO + X*80)/256):
E = LO + X*80-(D*256):GOSUB 200:NEXT
1830 LO = &HEBC0 + 65536!:POKE LO,141:
POKE LO + 79,142:H = &HEB:L = &HC1:B = 0:
C = 77:CH = 140:GOSUB 100
1840 LO = &HE857 + 65536!:FOR X = 1 TO 5:
READ E$(X):A$ = E$(X):D = INT((LO + X*80)/256):
E = LO + X*80-(D*256):GOSUB 200:NEXT
1900 FOR X = 1 TO PD:FOR Y = 1 TO 6:READ A:
READ A1:PRT$(X,Y) = CHR$(A) + CHR$(A1):
NEXT Y,X
```

```
1910 FOR X = 1 TO PD:READ PN$(X):NEXT
2000 FOR X = 1 TO 20:X$(X) = B$(X):NEXT
2005 A$ = "Header Selection: ":PP1 = 1525:
P1 = 567:P2 = 1:CO = 1:CL1 = 1367:CL2 = 1402:
SI = 20:NC = 35
2010 H = &HC0:L = 0:
GOSUB 900: 'Switch in screen 1
2020 S = 1:GOSUB 500: ' Get pointer - no F1
2030 LB(1) = P2:LB$(1) = B$(P2):LE(1) = 19:
' LB$(1) is the header
2040 IF P2 = 20 THEN GOSUB 3600:GOTO 2005
2100 H = &HC7:L = &H80:GOSUB 900: 'Screen#2
2105 PRINT@1858,ER2$;:' Get rid of prompt
2110 IF P2 = 19 THEN MC = LE(1):PRINT@80*11
 + 11,"Type your own: ";:FC = 906:GOSUB 300:
LB$(1) = A$:IF I$ = CHR$(129) THEN 2000
2200 IF P2 = 19 THEN P2 = 0:GOTO 2100
2205 GOSUB 400:PRINT@PL(1),LB$(1);
2210 LE(2) = 26:MC = LE(2):
PRINT@80*11 + 11,"Disk Name: ";
2220 FC = 902:GOSUB 300:LB$(2) = A$:
IF I$ = CHR$(129) THEN 2000
2230 GOSUB 400
2300 H = &HCF:L = 0:GOSUB 900: ' Screen#3
2310 FOR X = 1 TO 14:X$(X) = C$(X):NEXT
2320 FOR X = 1 TO 2:PRINT@PL(X),LB$(X);:
NEXT:LE(3) = 14
2330 A$ = "Operating System: ":PP1 = 1605:
P1 = 891:P2 = 1:CO = 1:CL1 = 1451:CL2 = 1482:
SI = 14:NC = 31:GOSUB 500
2340 IF I$ = CHR$(129) THEN 2100
ELSE LB$(3) = C$(P2)
2400 H = &HD6:L = &H80:GOSUB 900:'Screen#4
2410 FOR X = 1 TO 5:X$(X) = D$(X):
NEXT:PL(3) = 415
2420 FOR X = 1 TO 3:PRINT@PL(X),LB$(X);:
NEXT:LE(4) = 19
2430 A$ = "Disk Format: ":PP1 = 1605:
P1 = 987:P2 = 1:CO = 1:CL1 = 1387:CL2 = CL1:
SI = 5:NC = 0:FL = 1:GOSUB 500
2440 IF I$ = CHR$(129) THEN 2300
ELSE LB$(4) = D$(P2):GOSUB 400
2450 LB$(5) = DATE$:PL(5) = 458:LE(5) = 8
2500 H = &HDE:L = 0:GOSUB 900: ' Screen#5
2505 PRINT@1858,ER2$;: ' Get rid of prompt
2510 FOR X = 1 TO 5:PRINT@PL(X),LB$(X);:NEXT
2520 LE(6) = 38:MC = LE(6):FC = 912:GOSUB 300:
LB$(6) = A$:IF I$ = CHR$(129) THEN 2400
2530 GOSUB 400
2600 H = &HE5:L = &H80:GOSUB 900:'Screen#6
2610 FOR X = 1 TO 5:X$(X) = E$(X):NEXT
2620 FOR X = 1 TO 6:PRINT@PL(X),LB$(X);:NEXT
2630 A$ = "Make your selection: ":PP1 = 1685:
P1 = 802:P2 = 1:CO = 1:CL1 = 1202:CL2 = CL1:
SI = 5:NC = 0:FL = 1:GOSUB 500
2640 IF I$ = CHR$(129) THEN 2500
3000 ON P2 GOTO 3010,3200,3300,3400,3500
3010 PRINT@830,"Default printer =  ";PN$(PDF);:
MC = 3:FC = 1022:PRINT@990,"How many labels
would you like:     ";:GOSUB 300:
IF I$ = CHR$(129) THEN 2600 ELSE LN = VAL(A$):
IF LN THEN 3010
3020 MC = 5:FC = 1182:
PRINT@1153,"Type start ing serial number: ";:
PRINT@1233,"or press <ENTER> for none";:
GOSUB 300
3030 IF I$ = CHR$(129) THEN 2600 ELSE IF A$ =
CHR$(13) THEN LB$(7) = "" ELSE LB$(7) = A$:
LB = VAL(LB$(7))
3035 ' LPRINT routine
3040 FOR X = 1 TO LN
3050 FOR Y = 1 TO 2
3060 LPRINT TAB(2);PRT$(PDF,4);PRT$(PDF,1);
PRT$(PDF,6);STRING$(INT(((27-LEN(LB$(Y)))/2)),
32);LB$(Y);:NEXT
3070 LPRINT TAB(1);PRT$(PDF,2);LB$(3);
STRING$(INT(((48-LEN(LB$(3) + LB$(4) +
LB$(5)))/2)),32);LB$(4);STRING$(INT(((58-LEN
(LB$(3) + LB$(4) + LB$(5)))/2)),32);LB$(5)
3080 LPRINT TAB(1);STRING$(INT(((54-LEN
(LB$(6)))/2)),32);LB$(6)
3082 IF LB$(7) = "" THEN LPRINT" " ELSE LB$(7)
 = "Serial number: " + STR$(LB):LPRINT TAB(1);
STRING$(INT(((54-LEN(LB$(7)))/2)),32);LB$(7):
LB = LB + 1
3085 LPRINT PRT$(PDF,3):NEXT
3090 GOTO 2600
3200 LO = 832:FOR X = 1 TO 6:
PRINT@LO + X*80,LB$(X);:NEXT
3205 FOR X = 1 TO 6:X$(X) = LB$(X):
NEXT:PRINT@1685,STRING$(74,32);
3210 A$ = "Edit field: ":PP1 = 1685:P1 = 905:P2 = 1:
CO = 1:CL1 = 1385:CL2 = CL1:SI = 6:NC = 0:FL = 1:
GOSUB 500
3220 IF I$ = CHR$(129) THEN 2600
3230 PRINT@1685,STRING$(75,32);:
PRINT@1685,"Change to: ";:MC = LE(P2):
FC = 1696:GOSUB 300:IF I$ = CHR$(129)
THEN 2600 ELSE LB$(P2) = A$:
PRINT@823 + P2*80,STRING$(56,32);:
GOSUB 400:GOTO 3200
3300 PRINT@990,"Press <ENTER> to fill label
with asterisks ";
3310 I$ = INKEY$:IF I$ = CHR$(129) THEN 2600
ELSE IF I$ = CHR$(13) THEN 3320 ELSE 3310
3320 FOR X = 1 TO 6:LPRINT TAB(1);PRT$(1,3);
STRING$(34,"*"):NEXT:GOTO 2600
3400 LO = 834:FOR X = 1 TO PD:
PRINT@LO + X*80,PN$(X);:NEXT
3405 FOR X = 1 TO PD:X$(X) = PN$(X):NEXT:
PRINT@1685,STRING$(74,32);
3410 A$ = "Select printer: ":PP1 = 1685:P1 = 905:
P2 = 1:CO = 1:CL1 = 825 + 80*(PD + 1):CL2 = CL1:
SI = PD:NC = 0:FL = 1:GOSUB 500
3420 IF I$ = CHR$(129) THEN 2600
ELSE PDF = P2
3430 GOTO 2600
3500 GOSUB 3600:GOTO 2600
3600 PRINT@1685,"REBOOT DOS - Please press
Y to confirm";
3610 I$ = INKEY$:IF I$ = "" THEN 3610 ELSE IF
I$ = "Y" OR I$ = "y" THEN A = &H1BF2:CALL A
ELSE RETURN
```

Model 4 (model III mode)

Hidden Memory Fun

by Donald G. Shelton

The article showing how to page back and forth from the 64 x 16 to 80 x 24 modes on a Model 4 operating in Model III mode raised some interesting programming possibilities (TRSTimes July 1988, page 26). One that might not be thought of right away is that the second page of video is protected memory, just waiting for you to use it! The best application I thought of was simulating the COMMON statement, which isn't available in Model III basic. LDOS Basic has a function for saving all variables when you run a new program, but your choice is all or nothing, and this may cause problems when you only want to transfer a few key variables (such a the name of the data file being worked on).

The COMMON statement (which IS available in Model 4 BASIC) protects enough memory to keep variables you specify when you call one program from another. This is very handy in memory-limited situations, because it allows easy overlaying from disk of different program modules, yet keeps the variables you need as if you were running the program without overlays.

Without the COMMON statement you can still preserve memory and manually protect variables in it, but when working from BASIC this requires imbedding some machine language routines and calling them, which works well, but isn't very convenient (see Lewis Rosenfelder's "Basic Faster & Better").

This new method involves only calling a subroutine, printing the variable at a preset location, and making it no longer than some preset length (the receiving program needs to know where to look for the protected variable, and how long it is), all without leaving BASIC.

It is based on the fact that Model III video is memory mapped. This allows a technique called string "pointing". Normally strings are stored in the memory stack away from video memory, but you can point a string to a video memory location using the VARPTR statement. This means that you can make a string from bytes on the video display. What we are going to do is print a variable at a location on page 2 of video, run a new program, "point" a string

to that video location ("pointing" will work on either page of video) and recover the variable.

Things to know before we start: 1) the second program must know where the print@ locations are for each variable. 2) The second program must know the maximum length the variable might be. 3) the second program must recover the variables before it initializes any variables of its own, and before it does any printing to the second page of video.

You could decide on a standard length for strings (say 20 bytes), and simply locate them at positions 0,20,40,60, etc. If you don't know how many variables there will be, or you want the subroutine flexible among different programs, make the last variable "END" and test for it, just like you would in a data statement. You could change this routine for accepting integers and other numbers, or you could keep the routine flexible by converting numbers to strings before you store them, and back to numbers in the new program using STR$() and VAL().

The test programs demonstrating this technique are X/BAS and Y/BAS. X/BAS is extremely simple. All it does is accept a variable that you type in. Then in line 40 the OUT 132,128 statement puts the computer in a mode where it is writing to the second page of video, but still looking at the first page. This makes the process seem invisible to the user. It then prints the variable at position 0 on the second page of video. The computer tells you what it is doing at each step, and delays a few seconds (otherwise the whole thing would be over in less than a second). Then the program runs program Y/BAS. All variables are normally lost at this point.

In program Y/BAS A1% is set to 20, and is the maximum length of the variable. PO% is set to 0, and is the position of the variable on the second video page. The OUT 132,128 once again sets the computer to using the second page of video while the user only sees the first. Then it uses the "pointer" routine to create a variable AN$ which is actually stored on the second video page. At this point the program sets A$ equal to AN$. AN$ is not a "permanent" string. It points to the first 20 bytes of video memory, and as soon as we switch back to the first page of video, the contents of AN$ will change with it! That is why the contents of AN$ must immeadiately be transferred to a "normal" string, or you will lose it. Then an OUT 132,0 returns the screen to normal Model III mode, and A$ is printed for all to see.

A$ is now 20 bytes long, even if it was less when it started. This is similar to using fields with random access files; we had to determine a length before we started. At this point you may need to strip the string of its trailing blanks. I include a function for doing that (again thanks to Rosenfelder).

This is one of the cleanest memory banking 'tricks' I've ever seen for BASIC, and there are certainly other ways to use this valuable memory. By the

way, if you are running a program in the 80 x 24 mode, you can still use this. All you have to do is print the variables to locations on the second video page, switch to the first video page, run the second program, recover you variables from the second video page, and go on your way. Of course, with this method the process is no longer invisible to the user, but that may not be important.

X/BAS

```
0 'X/BAS ENTERS THE VARIABLE & STORES
IT ON SECOND VIDEO PAGE
(C) 1988 D.G. SHELTON - LEXINGTON, KY

10 CLS: LINE INPUT"ENTER THE STRING ";A$

20 A1% = LEN(A$)

30 PRINT: PRINT"I AM NOW PUTTING IT IN
EXTRA VIDEO MEMORY"

35 FOR D = 1 TO 2000: NEXT D

40 OUT 132,128: CLS

50 PO% = 0: PRINT@PO%,A$

60 OUT 132,0: PRINT:PRINT "DONE"

70 PRINT "NOW I WILL RUN ANOTHER
PROGRAM": FOR D = 1 TO 2000: NEXT D
```

Y/BAS

```
0 ' RECOVERS VARIABLE FROM SECOND VIDEO
PAGE (C) 1988 D.G. SHELTON

5 CLEAR 400

10 CLS: PRINT"I AM NOW RUNNING A NEW
PROGRAM. I WILL ACCESS VIDEO MEMORY":
PRINT"FOR THE VARIABLE":FOR D = 1 TO 2000:
NEXT D

15 DEFFNSS$(AN$) = LEFT$(AN$ + " ", INSTR
(AN$ + "  "," ")-1) 'FUNCTION FOR STRIPPING
TRAILING BLANKS

20 A1% = 20: PO% = 0

30 OUT 132,128: GOSUB 60: A$ = AN$:
OUT 132,0

35 A$ = FNSS$(A$)

40 PRINT: PRINT: PRINT"THE VARIABLE IS - ";A$

50 END

60 AN$ = " ": POKE VARPTR(AN$),A1%: POKE
VARPTR(AN$) + 2,INT(PO%/256) + 60: POKE
VARPTR(AN$) + 1,PO%-INT(PO%/256)*256:
RETURN
```

PUBLIC DOMAIN PROGRAMS

NEW PROGRAMS

from the Valley TRS-80 Hackers' Group
public domain library
for Model I, III & 4

Send SASE for annotated list

Sample disk $5.00 (US)

VTHG

BOX 9747

N. HOLLYWOOD, CA. 91609

MORE GOODIES FOR YOUR TRS-80

Get the latest issue of
TRSLINK
TRSLINK is the new disk-based magazine dedicated to providing continuing information for the TRS-80. A new issue is published monthly, featuring Public Domain programs, "Shareware", articles, hints & tips, nationwide ads, letters, and more.
TRSLINK can be obtained from your local TRS-80 BBS, or download it directly from:

8/n/1 #4
215 848-5728
(Philadelphia, PA.)
Sysop: Luis Garcia-Barrio

Believe it or not:
TRSLINK is FREE

xT.CAD: Computer Aided Drafting for the TRS-80

A Review by Eric Bagai

This review took a lot longer to write than I'd expected. Ordinarily, one has some familiarity with the category of any program to be reviewed. You expect to understand the various microcomputer metaphors of "spreadsheet," "word processor," "general ledger," or what-have-you. As a last resort you can always compare it to another program that does the same kind of thing.

xT.CAD is different.

In the TRS-80 world, xT.CAD stands alone. It is not only without peer, it is without competition of any kind. It also isn't like anything I've used before. That meant I had nothing to compare it to. My understanding depended on building myself a metaphor for Computer Aided Drafting (CAD).

The obvious place to start, I thought, was with a more familiar metaphor: high-resolution drawing programs. In drawing programs what you see is, well, what you see. That is, what you produce on the screen is reproduced, with little or no distortion, on paper. You get 640 by 240 pixels (less than 100 dpi), on the screen or on paper. But in xT.CAD, what you see on the screen is a function of the current level of magnification (among ten levels), and the result of an unlimited number of possible overlays. What you get on paper is a result of xT.CAD's internal accuracy of 1/200 of an inch, and depends how big your printer is. If your printer is a plotter then all your diagonal lines will loose their jagged edges and become absolutely straight lines.

But please don't mistake xT.CAD for a jazzed up drawing program. The comparison makes for great "gee-whiz" numbers, but it is not really very useful. xT.CAD operates in a basically different manner than do drawing programs. To begin with, in xT.CAD you don't really draw on the screen at all, you produce a series of notations that are represented on a field of from 8 by 10 inches to 24 by 36 inches-- the screen acts as a variable sized window on this field. Every drawing in xT.CAD begins in the top left corner of the screen, and every move from that position is recorded along with every line drawn or erased until the drawing is ended. At any time during construction the notation and the screen can be "played back" line by line (or automatically), to find the point in the notation at which an item was constructed, and can then be edited. The complete notation history is saved as an ASCII file. When that file is loaded the result of the history is redrawn on the screen.

The reason for this programming approach has to do with CAD's use and history versus microcomputer drawing's use and history. Although drawing programs will sketch or paint on the screen much as a artist would on canvas or paper, they are all a direct descendant of Etch-a-Sketch simulations. With the addition of block manipulation, variable brush size, fill patterns, zoom, mirror, complement, and reverse, along with access to hardware such as mice and digitizers, drawing programs have become a useful addition to the artist's toolbox. Even though most artists begin by making rough shapes, drawing on paper or screen becomes a non-linear process after they've done the first sketching. The artist moves from place to place, adding, subtracting, refining, etc., pixel by pixel. A finished drawing may light more than 90% of the screen's pixels, especially if the background is white.

For most of its history, CAD has been limited to minicomputers and high-end workstations. It quickly became accepted that, so long as data files were interchangeable, the screen resolution was not significant. At the lowest levels of screen resolution CAD is impractical because the screen is too small a window on the drawing; other than that, there is no reason why you couldn't design a CAD program for a Model 1 or a Model 100. xT.CAD became available for the TRS-80 in 1984, shortly after CAD appeared for the IBM-PC.

Drafting is a linear process: you know exactly what you want before you begin, and you often plan the order and progression of your work in some detail. CAD programs are used to represent physical objects in a manner precise enough so that they may be constructed from the information given in the drawing. All lines are drawn as if they were mathematical abstractions, having no thickness; the conventions of drafting require only a variety of solid, dotted, dashed, and dot-dashed lines. There is no such thing as an isolated point except to mark the beginning or end of a line segment; there are only lines.

Because all these notations must be held in memory, there is a definite limit to how much detail can be represented in one file. xT.CAD keeps track by counting each line/arc/circle as an item. A status line is always available to show how many items are free out of the 1000 items allotted per file. So what happens when you run out of available items? You use screen number two. Then you can switch between screens and even merge them. What's more,

you can use as many screensful as needed, with all drawings kept in register. This means you can produce the most complex of drawings with xT.CAD. These overlays can not only be shown simultaneously on the screen, they can also be plotted (or printed) over each other until the drawing is complete.

The value and speed of xT.CAD is increased with every drawing made, because each job establishes more items in your symbol library. After a short time you find that most of your work consists of using block moves to plug in items from your own library.

Now that I've made all these distinctions, I should confess that xT.CAD can also be used to advantage as a drawing program. For example, each overlay can be used with a different color printer ribbon to produce full-color drawings. Also, xT.CAD's dot matrix printer drivers (but not plotter drivers) allow you to fill areas with ten degrees of shading. And if you have an "E" size drawing but only an "A" sized plotter, or only an 80-column dot matrix printer, you can specify percent of reduction at print time.

If you've ever wanted to do any kind of drafting at all but felt you were a total klutz with pencil and straight-edge, then xT.CAD is for you. If you are an artist experienced with drawing programs you will find xT.CAD interesting for the possibilities it allows. If you are an engineer, planner, designer, or contractor of any stripe, then xT.CAD is worth investigating. To use xT.CAD you need a Model III or 4 and a high resolution board. Either Tandy or Microlab's high resolution boards will do. (If you insist, Microdex will sell you their IBM-PC version, but rest assured that it works just as well as the TRS-80 versions, and the data files are perfectly compatible.) It is fun to work with, easy to learn, and has many more features than I've mentioned here (like a complete Bill of Materials system, full ASCII labeling with automatic duplication, and digitized and coordinate input.) It is also an amazing piece of programming. But if you are a draftsman of any degree of ability or experience in any field, you'll find that xT.CAD will meet your most exacting needs.

xT.CAD was developed by Microdex. Write or call them at:

Microdex Corp.
1212 N. Sawtelle
Tucson, AZ. 85716
(602) 326-3502

It is also available from Microlabs along with Model III and 4 Graphics Solution high-resolution boards and software. Both companies have large selection of high-resolution programs, boards, and accessories for the TRS-80, and their ads appear in each issue of TRSTimes.

TRSTimes on DISK #2

Issue #2 of TRSTimes on DISK is now available. It features the following programs from the July, September an November 1988 issues:

STATES/BAS	M4	TRSDOS 6.2 & 6.3
MASTRKEY/CMD	M3	LDOS 5.1.3
MASTRKEY/CMD	M4	TRSDOS 6.2 & 6.3
CHANGO80/BAS	M3/4	ALL
SCRPATCH/BAS	M3	ALL
MACROKEY/CMD	M3	NEWDOS/80
TRSTEXT2/BAS	M4	TRSDOS 6.2 & 6.3
TRSDRAW/BAS	M4	TRSDOS 6.2 & 6.3
WINDOW/BAS	M4	TRSDOS 6.2 & 6.3
LABEL204/BAS	M4	TRSDOS 6.2 & 6.3
X/BAS	M3/4	ALL
Y/BAS	M3/4	ALL
PLOTZ/BAS	M3	ALL

Included on this disk will be two assembly language programs that are just too long to publish in TRSTimes:

LABELDMP/CMD is the Model III parent of LABEL204/BAS. Works with the DMP series printer.

FORMAT4/CMD is a fast Model 4 format utility for TRSDOS 6.2 & 6.3. Works with 40 track, single sided diskettes only.

TRSTimes on DISK #2 is reasonably priced:

U.S. & Canada: $5.00 (U.S.)

Anywhere else: $7.00 (U.S.)
shipped air-mail

Send check or money order to:

TRSTimes on DISK
20311 Sherman Way #221
Canoga Park, CA. 91306
U.S.A.

TRSTimes on DISK #1
is still available at the above price.

Model III

PLOTZ

PLOTTING ON THE PRINTER

by Jim E. King, MSEE, C10

In 1974 I studied during the summer at Cal. State Northridge on plotting on the printer to show stock market time series data and came up with a Fortran version of this routine. This is just one of many plotting routines.

The video display routine is from lines 300 to 350 with subroutines from 200 to 252. For printout, the routine is from 400 to 450. The two routines are almost identical when 80 column screen and paper are used.

Definitions of variables for the algorithm:

DL Lowest datum = Y(min), line 210

DH Highest datum = Y(max), line 212

R Range of the data = DH - DL, line 214

IP The # of the column where the datum is printed, line 220

IZ The # of the column where the zero line is printed, line 216

KC Coeficient used in the calculation of IP derived from the # of columns used, = # of columns -2, lines 120, 130, 220

N Number of data, line 110, 320, 420

Y(I) Data in array Y(I), lines 92, 210

Z0 = "0" zero plotting symbol, line 250

ZH Heading, lines 92, 320, 420

ZL The line # in $tring form left justified, line 230

J Length of line # - 1, line 230

ZP = "*" curve plotting symbol, line 94. Any symbol can be used.

The routine first finds the largest (DH) and smallest (DL) data, and the Range (R) between them by (GOSUB 210).

It then prints DL, the heading, the resolution, and DH.

The resolution of a printer for plotting is pretty poor, so to get the best available I scaled and positioned the data so that it goes from the leftmost column to the rightmost, tab(0) to tab(79), 79 intervals. This is accomplished in subroutine 220.

The following triangle diagram assists in understanding and setting up the ratio equations for converting the data to plotting positions:

Data	Tab(#)	Distances set up as ratios:
DH = Y(max)	79	IP-0 = Y(I) - Y(min)
		79-0 = Y(max) - Y(min)
Y(I)	IP	

Range = DH - DL

Solving for IP:

(1) IP = 79*(Y(I)-Y(min))/Range

DL = Y(min),0

Since IP is an integer number (DEFINT I-N) rounding must be introduced by adding 0.5 to the right side of the equation. Equation (1) is implemented in line 220, where KC = 79 - 1.

Tabs on our computers go from 0 on the left side, to 79 on the right, for an 80 column page or screen. Placing a character in tab(79) forces a linefeed on the screen but does not on my printer (Citoh), therefore KC must be reduced from 79 to 78 (line 120) for video, but not for printout. This is a slight difference between the Video routine and the Printout routine.

Subroutine line 230 converts the integer line numbers of the data to $trings with no left blank and is used in lines 330 and 430. The IF IP > J the plot print over the line #s.

IZ is the place on the plot where the variable goes to zero. It can be displayed to assist in getting a feel for the centering of the data about zero (line 216). Turning the "0" off is done by setting it = "".

It is possible for the "0" line to be outside the Range, i.e. off the page. A negative tab causes an Illegal Function error and everything stops, so the first IF in line 340 checks if IZ is off scale and if it is then prints ZP at IP.

Because it is not possible to tab backward, it is necessary to determine whether IP or IZ is smaller, and then print the smaller value first.

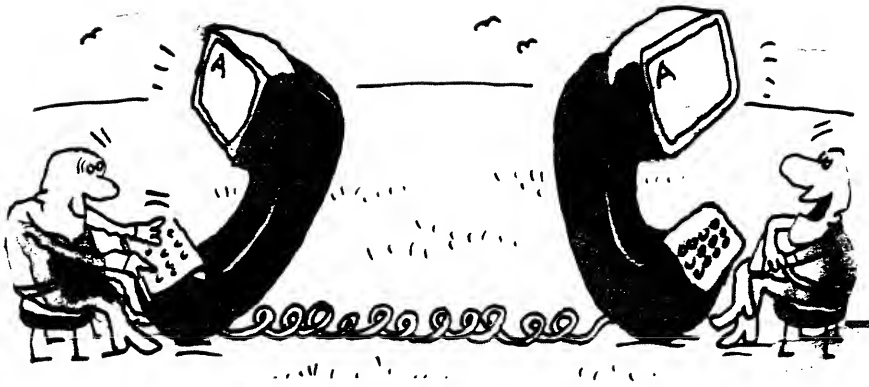
A Fortran version that will print 9 variables is available for a business size SASE to:

JIM KING

20784 Medley Street. Topanga, CA. 90290

PLOTZ/BAS

```
0 CLS:CLEAR555:DEFSTR S-X,Z:
S = "Demonstration of Plotting on Printer Routine,
With Zero Line":GOSUB 3:S = "by Jim E. King,
Copyright (c)(p) 1974":GOSUB 3:
S = "Public Domain; Permission to use if credit to
the author":GOSUB 3:GOTO 90  'PLOTZ

3 PRINT TAB(33-LEN(S)/2);S:RETURN' center

7 IF LEN(Z) THEN FOR L9 = 1 TO LEN(Z):
K9 = ASC(MID$(Z,L9,1)):IF K9 > 96 THEN
MID$(Z,L9,1) = CHR$(K9-32):NEXT:RETURN
ELSE NEXT:RETURN ELSE RETURN 'CAPS

8 Z = INKEY$:IF Z = "" THEN 8 ELSE IF Z =
CHR$(31) THEN END ELSE GOSUB 7:RETURN

9 PRINT CHR$(29);STRING$(JU + 1,27);
CHR$(31);:JU = 0:RETURN  ' jump up lines

30 PRINT ZH:FOR I = 0 TO N:PRINTI,Y(I):
NEXT:GOTO 99  ' display Data

80 PRINT ERR"ERROR-Line"ERL:GOTO 99

90  DEFINT I-N: M = 166: DIM Y(M): N = M:
KC = 78:ON ERROR GOTO 80

92 A = .6: B = .11: FOR I = 0 TO N: Y(I) = A +
SIN(B*I): NEXT:ZH = "Sine Wave:.6 + Sin(.11*I)"
'put sine data into array Y(I)

94 X = "OPEN Disk Drive Doors":ZP = "*"

99 PRINT"DIMension"M; TAB(40);KC + 2;
" < C > olumns  < 6 > 4/ < 4 > 0  < Z > ero";Z0;
" Line  < D > isplay";N;"Da < t > a  < P > lot
 < L > Plot?";:GOSUB 8:PRINT Z

110 IF Z = "N" ORZ = "T" THEN PRINT N;"Lines of
Data out of";M;"DIM; (61 lines/page) Change to";:
INPUT N:N = -(N < = M)*ABS(N):JU = 2:GOSUB 9

120 IF Z = "6" THEN KC = 62:JU = 1:GOSUB 9

130 IF Z = "8" THEN KC = 78:JU = 1:GOSUB 9

140 IF Z = "C" THEN INPUT"Change # of
Columns to";K: KC = K-2

150 IF Z = "Z" THEN GOSUB 250 ELSE IF Z = "D"
THEN 30 ELSE IF Z = "P" THEN 300 ELSE IF Z =
"L" THEN 410

190 GOTO 99

200 'Plotting Subroutines

210 DH = -9E9: DL = 9E9: FOR I = 0 TO N:
IF D > LY(I) THEN DL = Y(I)  ' DL = Y(min)

212 IF DH < Y(I) THEN DH = Y(I)  ' DH = Y(max)

214 NEXT: R = DH-DL  ' range of data

216 IZ = -KC*DL/R + .5: RETURN
' calc "0" by triangle ratio where Y(0) = 0

220 IP = KC*(Y(I)-DL)/R + .5:RETURN
' calc tab(IP) by triangle ratio

230 ZL = RIGHT$(STR$(I),2):RETURN
' 2 digit line# -- > $

240 ZL = RIGHT$(STR$(I),3):RETURN
' 3 digit line# -- > $

250 IF Z0 = "" THEN Z0 = "0" ELSE Z0 = ""

252 JU = 1: GOSUB 9:RETURN' toggle '0' line

300 'Video Plot

320 CLS:S = ZH:GOSUB 3:GOSUB 210:
PRINTDL" Resolution  = ";R/(KC + 1);
TAB(KC-8);DH:FOR I = 0 TO N:GOSUB 220

330 IF I < 100 THEN GOSUB 230:IF IP > 2 THEN
PRINT ZL; ELSE ELSE GOSUB 240:IF IP > 3 THEN
PRINT ZL;  'left side line#

340 IF IZ < 0 OR IZ > KC THEN PRINT TAB(IP);ZP
ELSE IF IP < IZ THEN PRINT TAB(IP);ZP;TAB(IZ);
Z0 ELSE IF IP = IZ THEN PRINT TAB(IP);ZP ELSE
PRINT TAB(IZ);Z0;TAB(IP);ZP ' display the points

350 NEXT: GOTO 99'ZP = Plotting Symbol;Z0 = "0"

400 'LPrint Plot

410 K = 80:INPUT"Turn on Printer;  # Columns
(80)";K: KC = K-1

420 GOSUB 210:
LPRINT DL;" ";ZH;" Resolution  = ";R/(KC);
TAB(KC-8);DH: FOR I = 0 TO N: GOSUB 220

430 IF I < 0 THEN GOSUB 230: IF IP > 2 THEN
LPRINT ZL;
ELSE ELSE GOSUB 240:
IF IP > 3 THEN LPRINT ZL;  'left side line#

440 IF IZ < 0 OR IZ > KC THEN LPRINT TAB(IP);
ZP ELSE IF IP < IZ THEN LPRINT TAB(IP);ZP;
TAB(IZ);Z0 ELSEIFIP = IZ THEN LPRINT TAB(IP);
ZP ELSE LPRINT TAB(IZ);Z0;TAB(IP);ZP
' print the points

450 NEXT: KC = 78: GOTO 99
```



Tim's PD Express

The File Cabinet update

by Timothy Sewell

I NEED YOU HELP!

This issue's PD Express is going to be a bit short due to the write up and code of LABEL204/BAS found elsewhere in this issue. I just want to alert you to a few things that have happened in the last few weeks.

The big news is that THE FILE CABINET HAS A NEW ADDRESS. The postal service in it's infinite wisdom decided to close down the Post Office where I received my mail and build a larger one on the other side of San Fernando. Since I moved away from San Fernando a couple of months back, I decided to apply for a P.O. Box closer to where I lived instead of using the new Box number that was being assinged to me at the new facility. So, since I was going to have to change box numbers anyway it might as well be one closer to home.

So please make note of The File Cabinet's NEW address:

The File Cabinet
P.O. Box 322
Van Nuys, Ca. 91408

All mail sent to the old address will be forwarded to my new one so don't worry. It just may take a couple of days longer to get to me.

The other news is that there has been A MAJOR UPGRADE TO THE MODEL 4 CATALOG.

I have added close to 40 new disks to the library and have updated several files in the existing library. To obtain an updated catalog, send me your ORIGINAL catalog disk(s) in a mailer with RETURN POSTAGE and MAILING LABEL and I will copy the new catalog and send it right out to you.

The Model 1/3 catalog has been delayed. The project has become so huge that I am not able to meet the September deadline like I promised. I am targeting an early December release. For those of you who pre-paid your Model 1/3 catalog and don't want to wait that long, I will be happy to refund your deposit. Just write me and let me know.

The Orchestra-90 Music file catalog WAS finished until a couple of days ago when John Davis (thanks John) sent me 5 diskettes full of new files to add to the library. It will be a couple of more weeks before the catalog is ready to be shipped while I sort out the new files.

The largest problem with putting together the Model 1/3 catalog is trying to decide what is Public Domain and what isn't. If you have any of the following you would be interested in parting with (for a price or diskettes from the catalog) I would be really interested in hearing from you:

CLOAD Magazine on cassette or disk. I also need to know the dates of the first and last issue.

SOFTSIDE Magazine. Any printed magazine before September 1980 or after February 1984.

Any disk magazine before 9/81 and after issue #33. Also any special edition (best of) magazines and disks

80-NW Magazine. Volume 1 #1 only.

FAMILY COMPUTING Magazine. Volume 1 #1 through volume 1 #12 (entire first year).

80-US magazines on DISK.

Any software company's catalog for TRS-80 software.

Any **Computronics magazine** before 1984.

All issues of the **MISOSYS Quarterly** (Originals only please!).

Any **journals from the various DOS systems** available (LDOS, DOSPLUS, etc.).

If you can help with any of these items I would very much appreciate it. Feel free to write to me via The File Cabinet.

Next issue: I will be starting off the second year of TRSTimes with my report on telecommunications software available for the Model 4 and the Model 3.

I will also be reviewing BBS programs in the next year for those of you who have the notion of setting one up. Also look for my article on the experiences of setting up the CP/M version of Printmaster Plus on the Model 4. And I promise... more information about Model 1/3 software.

Until then...
Make Mine TRS-80!

ITEMS OF INTEREST

HARDWARE

XLR8 BOARDS

WITH memory chips AND docs

$200.00 (just a few left)

LNW DOUBLERS

without 1791 chip for
LNW & Model I.

$10.00 ea.

MODEL I DOUBLERS

NEW
In original box
NEVER USED
with binder of docs. RS#26-1143

$25.00 ea.

AZTEC 65 WATT POWER SUPPLIES

NEW - In easily removable shield case.
Originally used in Model 4/4P
until Radio Shack
substituted them with cheaples.

$20.00 ea.

Please add $5.00 shipping & handling
for each item.

JACK EICH
1643 BOLINGRIDGE DR.
ORANGE, CA. 92665
(714) 637-2943

SOFTWARE

TRS-80 SOFTWARE for Models 1/3/4/4P/4D

Many useful programs - Economical prices
Send $2.00 for listing

PRACTICAL PROGRAMS
1104 ASPEN DRIVE
TOMS RIVER, N.J. 08753

The RAM software Company presents:

SMALL-C compiler version 3.0

on the TRS-80

A large subset of Kernighan and Ritchie C, with a
UNIX compatible I/O library.
Many other library functions are included.
This is a true compiler, not a psuedo-code generator
like some others.

REQUIRES that the purchaser own
Microsoft's M80 assembler and L80 linker,
or compatible assembler and linker,
on a 48K Model I with Newdos/80 version 2.0.
More than 1 disk drive is recommended.

$20.00 for executable C compiler,
library object code,
demo programs source code
and C manual.

$20.00 additional for
source code to compiler & library
and library building/management utilities
+ documentation.

Make Checks/Money orders payable to:

Ben Mesander
1137 E. Brooks St. Apt.4
Norman, Oklahoma. 73071

Sorry, no COD or credit cards.

TRS-80 is a trademark of Tandy Corporation
UNIX is a trademark of Bell Laboratories
NEWDOS/80 2.0 is a trademark of Apparat, Inc.

The File Cabinet is MOVING!

They said it would never happen, but the Postal Service has decided to close down my Post Office so they can build s larger one. This means that I have to MOVE!

Please make note of THE FILE CABINET'S new address so you can keep on enjoying the advantages of DOWNLOADING THROUGH THE MAIL without delays!

Over the years, THE FILE CABINET has collected TRS-80 software from all over the country and has compiled them into the largest collection you will find ANYWHERE!

Each disk is filled to near capacity with Transmission Error Free software that saves you the hassle of long download times and HUGE PHONE BILLS!

A two disk catalog of TRS-80 MODEL 4 software is available for only $5.00 which is refundable with your first order. The HIGH RESOLUTION/READMAC catalog is only $4.00 which is also refundable.

ANNOUNCING THE FILE CABINET'S ORCHESTRA-90 MUSIC FILE CATALOG

Hundreds of music files are now available for use with your Orchestra-90!
The catalog is available for $2.00 which is refundable with your first order.

MODEL 1/3 CATALOG WILL BE RELEASED SOON! WATCH FOR THE ANNOUNCEMENT

Send your catalog requests to: THE FILE CABINET
P.O. Box 322
NEW ADDRESS: Van Nuys, Ca. 91408

The File Cabinet

DOWNLOAD THROUGH THE MAIL!

Graphics Solutions

High-Resolution Software and Hardware

GBASIC 3.0 - Radio Shack Model 4/4D/4P/III hi-res board owners take note of an enhanced graphics Basic: GBASIC 3.0 It not only provides an equivalent for each of the BASICG commands but adds a number of important new ones while using less memory. Without having to exit Basic, the hi-res screen can be saved to disk, loaded from disk, or printed on any of 30 popular printers: Epson, Star Micronics, Radio Shack, Okidata, C. Itoh, NEC, etc. The software works with TRSDOS 1.3, 6.1.2, 6.2, 6.3; Dosplus 3.4, 3.5. 4; LDOS; and NEWDOS80. The disk contains 40 graphics programs/files. Also included is a detailed manual with assembly language entry addresses. $39.95. (Specify Model 4 or III mode or add $10 for both.)

The following eleven programs run on a Model 4/4D/4P/III equipped with a Radio Shack graphics board and GBASIC 3.0 or a Micro-Labs Grafyx Solution board:

DRAW - A powerful full screen graphics drawing and editing program. $34.95.

BIZGRAPH - Create business graphs from hand-entered or VisiCalc data. $59.95.

xT.CAD - Professional drafting aid which outputs to a printer or plotter. $145.00.

SURFACE PLOT - Plot three-dimensional equations of the form Z=F(x,y). $39.95.

3D-PLOT - View three-dimensional data from any perspective or angle. $29.95.

MATHPLOT - Plot equations of the form Y=F(x) with auto scaling. $29.95.

CHESS - A very powerful program with 10 skill levels, 40 play options. $39.95.

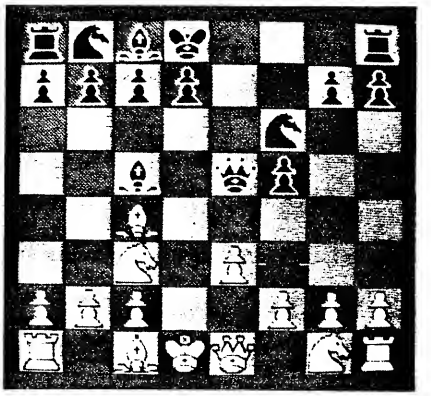
REVERSI - Play Othello with 10 skill levels, 20 execution options. $29.95.

3D Tic-Tac-Toe - Play the computer or a friend on a 4 × 4 × 4 matrix. $19.95.

SLIDESHOW - Create a sequence of hi-resolution picture displays. $19.95.

Biorhythm/USA - Plot your biorhythm or learn the states and capitols. $14.95.

JOY-MOUSE - Allows a Radio Shack Color Computer joystick, mouse, or touch pad to be connected to any Model 4/4D/4P/III. Hardware provides X, Y position values from 0 to 255. $119.95.



GRAFYX SOLUTION - A plug-in, clip-on board enhances any Model 4/4D 4P/III to provide 640 × 240 dot graphics. (512 × 192 on a Model III) The board comes with a 56 page manual and a disk containing both model 3 and 4 mode versions of over 40 programs and files including GBASIC 3.0 which adds over 20 graphics commands to Basic. $129.95.

Please specify your exact system configuration when ordering or requesting information. Payment may be by check, Visa, Mastercard, or COD. Domestic shipping is free on pre-paid orders. Texas residents add 7% sales tax.

MICRO-LABS, INC. 214-235-0915
902 Pinecrest, Richardson, Texas 75080

SOFTWARE

TRSDOS 1.4. PATCHER

The definitive enhancement to TRSDOS 1.3.

- allows access to double sided drives
- allows access to 8 drives (0-7)
- DIR command no longer requires the colon
- DIR command pauses between video pages
- Directory now uses full date stamping on new files
- DOS commands may be in upper, lower or mixed case
- DOS commands repeat by pressing <ENTER>
- DOS commands starting with a period are ignored
- DOS errors are expressed in full error messages
- LIST command defaults to ASCII
- All known HIT/GAT/DISK I/O errors have been corrected
- and much, much more

ONLY $29.95

DBSIDE
SUITE 209-1051 KLO ROAD
KELOWNA, BRITISH COLUMBIA
CANADA V1Y 4X6
(604) 762-8593

PUBLICATIONS

SUPPORT for your TRS-80

THE ONLY MONTHLY PUBLICATION THAT SUPPORTS YOUR MODEL I, III, IV, 4P & 4D

CONCENTRATION IS ON THE USER APPLICATION OF PROGRAMS, SOURCES OF PRODUCTS, PRODUCT REVIEWS, FEED BACK LOOP AND NEWS ITEMS FOR THE TRS-80 USER

$ 18.00 FOR FIRST YEAR INTRODUCTORY OFFER MAILED IN THE US
$ 29.50 CANADA AND MEXICO; $ 30.00 OUTSIDE THE US, CANADA AND MEXICO

Sample Issue $2.00 US And Canada

307-265-6483

Computer News 80

P. O. Box 680
Casper, Wyoming 82602-0680

TRS-80 Software from Hypersoft.

Read CP/M CoCo & PC disks on your TRS80

Use **HYPERCROSS** to COPY files between TRS-80 disks and those from many **CP/M** and **IBM-PC** type computers on your TRS-80 I, III, 4/4P or Max-80. You can FORMAT alien disks, read their directories, copy files to and from them, copy directly from one alien disk to another. Converts tokenized TRS80 BASIC to MSDOS or CP/M as it copies.
Formats supported: IBM-PC and MS-DOS including DOS 1.1, 2.0-3.2 Tandy 2000, single and double sided, 3.5 and 5 inch. CP/M from Aardvark to Zorba. CoCo format on XT+ version.
HyperCross 3.0 PC reads popular MSDOS 1.1-3.2 formats Order SX3PCM1, SX3PCM3 or SX3PCM4 .....$49.95
HyperCross XT/3.0 reads 90 different CP/M and PC formats Order SX3XTM1, SX3XTM3 or SX3XTM4 .....$89.95
HyperCross XT/3.0-Plus. Reads over 220 formats inc CoCo Order SX3XTM1+, SX3XTM3+ or SX3XTM4+ $129.95
Specify TRS-80 Model I (needs doubler), III, 4/4P or MAX-80. Dual model versions e.g. Mod 3/4 on one disk add $10 extra.

Amazing HYPERZAP 3.2G Disk Magic!

Do you want to backup, fix or modify a disk - if so then **you need HYPERZAP!**. More than just another disk copying program - it is the program for analyzing, copying, repairing, creating floppy disks of all kinds. It works with TRS-80 formats as well as many others such as CP/M, PC, CoCo etc. Designed to handle mixed density sectors on any track in any sequence. Many features for reading, writing, editing track and sector data. **Make your own self booting disks. Autopilot mode** learns, saves and repeats procedures. Disk comes with fascinating examples. Use Hyperzap as a learning tool, find how things are done!.
HYPERZAP 3.2G - nothing else even comes close! Order # HZ32 - one version runs on all Model I/III/4/4Ps $49.95

Other TRS-80 Programs

FORTH: Mod 1/3 $49.95, Enhanced Model 4 version: $59.95
LAZYWRITER Word Processor for Model 1, 3 or 4 .....$109.95
MultiDOS 2.1 1988 and beyond ! Model 1 or 3 .....$79.00
MultiDOS 2.1 64/80 version for Model 4 .....$89.00
Mysterious Adventures - Set of 10 for M1, 3 or 4(3) complete $49.95
NUTRITION Analyze your diet, with database, Model 4 only $49.00
TASMON debug trace disassemble TASM1 TASM3 or TASM4 $49.95
TMDD Memory Disk Drive for NewDOS 80/Model 4 users $39.95
XAS68K 68000 Cross Assembler, specify Mod 1, 3 or 4 .....$49.95
ZEUS Editor/Assembler specify Model 1, 3 or 4 ..... $74.00
ZIPLOAD fast load ROM image, DOS & RAMDISK on your 4P $29.95

Run Model 4 Software on a PC with PC-Four !

Now you can run your favorite TRS-80 Model 4 programs on a PC !. **PC-Four** is a program that makes your PC or Compatible **behave like a 128K TRS-80 Model 4 complete** with operating system, Z80 microprocessor that can run many true Model 4 programs such as **ALDS, ALLWRITE, BASCOM, BASIC, C, COBOL, EDAS, ELECTRIC WEBSTER, FED, FORTRAN, HARTForth, Little Brother, MULTI-BASIC, MZAL, PFS FILE, PASCAL, Payroll, PowerMail, PROFILE, SUPERSCRIPSIT, TASMON, VISICALC, ZEUS and more.**

Runs on PCs, PS/2s, compatibles and laptops with at least 384K of memory. ONLY emulates Model 4 mode of Model 4. To use it you must transfer your old files to MSDOS disks using **PCXZ or Hypercross.**

Prices: Order #PC4 $79.95 alone, #PC4H $104.95 with Hypercross SX3PCM4, #PC4Z $119.95 with PCXZ. Available on 3.5" disk format.

PCXZ reads TRS80 disks on a PC

PC Cross-Zap (PCXZ) is a utility that lets you copy files to or from TRS-80 disks on a PC or AT. Transfers BASIC, ASCII and Binary files. **Converts BASIC and text files automatically.** You can also format a disk, copy disks, explore, read and write sector data, repair bad directories and much more. **Supports:** all double density Model I, III and 4 formats. **Requires:** PC, XT, AT or compatible. **You must have at least one 5-1/4" regular or high density drive and 256K memory.** Not for PS/2s: **Order # PCXZ $79.95**

Hypersoft

POB 51155, Raleigh, NC 27609

Orders: 919 847-4779 8am-6pm, Support 919 846-1637 6pm-11pm EST

MasterCard, VISA, COD, Checks, POs. Add $3 Shipping, $5 2nd day

CLOSE#6

Whew! This was another tough one. Same reason as last issue: Too much material. Though this is the largest issue we have produced to date, in order to achieve a good blend, it was again necessary to shuffle articles to upcoming issues. The cross-referencing utility for Visicalc, which was promised for this issue, WILL appear in January. Also upcoming will be a nice NX-10 printer utility by Danny Mullen, more articles by Fred Blechman, as well as John Fowler's fine bit of 'hackers' information about transferring Infocom adventure games from IBM to fully WORK on the Model III and 4.

The January issue begins our second year and, though right now it seems far away, we better start preparing for the assembly language tutorial.

I promised to make it as easy as possible for someone who is fairly comfortable using Basic. That promise stands. However, to make things reasonably managable for myself, I will confine the tutorial to Model III, TRSDOS 1.3. and Radio Shack's disk version of EDTASM.

This will obviously simplify the task greatly. All Model 4 owners have a built in Model III, everyone should have a copy of TRSDOS 1.3., and the simplest editor/assembler to work with is EDTASM. Trying to cover Model I, III & 4, all the various versions of applicable DOSes, as well as the many assembler packages available, would result in long, drawn-out mess that wouldn't teach you anything. Therefore: Model III, TRSDOS 1.3. and EDTASM.

If you don't have either TRSDOS 1.3. or EDTASM, **get them**. As the tutorial moves on, we will try to cover the Model 4 concept of Supervisor Calls (SVC).

The January issue will also see a regular column devoted to our Model I friends. It will be written by Ben Mesander, a talented and devoted TRS-80 user.

Model III and 4 will get regular coverage, and we do have some surprises coming up. Tim, Roy and Eric will be back with their usual high standard of bringing us information.

So, if you have not yet subscribed to the six 1989 issues, don't waste any more time, do so now.

Meanwhile, for making issue #6 possible, TRSTimes extends sincere thanks

- to Eric Bagai, president of the Valley Hackers Group, for The Real Hackers and xT.CAD: Computer Aided Drafting for the TRS-80.
- to George Madison, resident SAGATUG HI-Rez genius, for TRSTEXT2.
- to Robert Doerr for sharing his fine article: WINDOWS IN BASIC.
- to Roy Beck, member of every computer club in Greater Los Angeles (according to Barbara), for another great tutorial on CP/M.
- to Tim Sewell, the Guru of GEnie, for LABEL204 and The PD Express.
- to Donald Shelton for taking a concept and bringing it further to give us Hidden Memory Fun.
- to Jim King, the electricity expert of the Hackers' Group, for PLOTZ. (make that PLOT Z).

In closing, we would like to thank YOU, the readers. Without you, none of the first six issues would have been possible. You encouraged and supported our effort; you forgave us when we made mistakes; you shared your knowledge and programs. You have been great.

Thank you all for a wonderful first year.

Lance W.

BULLETIN BOARDS

- **Illinois**
- **Chicago**

CHICAHO SYSLINK NETWORK
(312) 622-4442
300/1200 baud (24 hours)
Sysop: George Matyaszek
Supports: TRS-80 and others.

Kentucky
Lexington

THE LEX80 BBS
(606) 268-0760
300/1200 baud
Weekdays: 6:00 pm. to 8:00 am. EDT
Weekends: 24 hours
Sysop: Donald Shelton
Supports: TRS-80

BBS CORRECTIONS

In the September 1988 issue, page 29, we gave an incorrect phone number for the KINGS MARKET BBS in Denver, CO. The correct phone number is: (303) 665-6091

Sorry about that.

Also, same issue, same page, the T.B.B.S (R)emote is no longer on line.